本科学生毕业设计

基于协同推荐的视频网站的设计与实现

系部名称： 计算机科学与技术学院

专业班级： 计算机科学与技术17-4

学生姓名： 陈盈涛

指导教师： 杨泽雪

职 称： 副教授

**黑 龙 江 工 程 学 院**

二○二一年六月

The Graduation Design for Bachelor's Degree

Design and Implementation of Video Website based on Collaborative Recommendation

Candidate：Chen Yingtao

Specialty：Computer Science And Technology

Class：17-4

Supervisor：Associate Prof. Yang Zexue

Heilongjiang Institute of Technology

2021-06·Harbin

摘 要

推荐系统作为大数据盛行的产物，是应对数据过载问题的产物，可以帮助解决信息消费用户和信息提供者之间的信息准确、快速匹配问题。推荐系统可以给网站带来直接的流量和经济效益，因此当前世界著名的网站平台都在发展推荐系统。

视频推荐系统一个通过搜集用户的日常行为，挖掘信息背后的用户个人偏好，进行物体分类的分类系统。根据分类对象可以分为，用户和项；用户的行为数据可以是历史记录、评分信息、搜索点击次数、评论等。推荐系统最核心内容是如何分类，协同过滤推荐算法按照分类的对象可以分为，基于用户的协同过滤和基于项的协同过滤。前者构造用户相似性矩阵，后者构造项的相似性矩阵。

本设计通过应用协同过滤推荐算法建立一个个性化视频网站，能够根据不同用户建立个性化电影推荐清单，设计过程中设计到基于用户的协同过滤、基于项的协同过滤的算法实验过程；使用Flask Web架构设计视频网站过程；以及利用NoSql缓存数据库优化推荐系统的效率过程。

关键词：协同过滤；视频网站；推荐系统；MovieLens；Flask；NoSql

ABSTRACT

As the product of big data, recommendation system is the product of dealing with the problem of data overload. It can help solve the problem of accurate and fast information matching between information consumers and information providers. Recommendation system can bring direct traffic and economic benefits to the website, so the world-famous website platforms are developing recommendation system.

Video recommendation system is a classification system that collects users' daily behaviors, mines users' personal preferences behind information, and classifies objects. According to the classification object, it can be divided into user and item; User behavior data can be historical records, rating information, search hits, comments, etc. The core content of recommendation system is how to classify. According to the classification objects, collaborative filtering recommendation algorithm can be divided into user based collaborative filtering and item based collaborative filtering. The former constructs user similarity matrix, and the latter constructs item similarity matrix.

This design builds a personalized video website by applying collaborative filtering recommendation algorithm, which can build personalized movie recommendation list according to different users; Use the website design architecture process under the flash web architecture; As well as using NoSQL cache database to optimize the efficiency of the recommender system.

**Key words**: Collaborative filtering; Video website; Recommendation system; MovieLens

Flask；NoSql

目 录

[摘 要 III](#_Toc13237)

[ABSTRACT IV](#_Toc31822)

[目 录 1](#_Toc23747)

[第1章 绪 论 4](#_Toc21270)

[1.1 研究背景 4](#_Toc5556)

[1.2 研究目的和意义 5](#_Toc22281)

[1.3 研究现状 6](#_Toc18862)

[1.3.1 推荐算法发展 6](#_Toc5955)

[1.3.2 推荐算法的研究现状 6](#_Toc8613)

[1.3.3 国内外推荐算法应用现状 6](#_Toc15886)

[1.4 本文安排 7](#_Toc14166)

[第2章 相关技术总览介绍 8](#_Toc7267)

[2.1推荐算法总览 8](#_Toc1346)

[3.1.1基于内容的推荐 8](#_Toc8832)

[3.1.2 基于内存的协同过滤 8](#_Toc9063)

[3.1.4 基于模型的协同过滤 8](#_Toc13887)

[3.1.5 混合推荐算法 9](#_Toc7872)

[2.2 相似度算法总览 9](#_Toc7440)

[3.2.1 欧几里得距离 9](#_Toc27099)

[3.2.2 曼哈顿距离 10](#_Toc7868)

[3.2.3 切比雪夫距离 10](#_Toc5890)

[3.2.4 闵可夫斯基距离 10](#_Toc28517)

[3.2.5 余弦相似度 10](#_Toc20683)

[3.2.6 杰卡德相似度 11](#_Toc8796)

[2.3 推荐系统评测方案 11](#_Toc27661)

[3.3.1 离线实验 11](#_Toc13504)

[3.3.2 用户调查 11](#_Toc1669)

[3.3.3 在线实验 12](#_Toc30215)

[2.4推荐算法评测指标 12](#_Toc32700)

[2.4.1混淆矩阵及其相关指标 12](#_Toc1504)

[2.4.2 ROC曲线和AUC面积 13](#_Toc17888)

[2.4.3 F分数 14](#_Toc6668)

[2.4.4 评分预测评价指标 14](#_Toc32436)

[2.5 本章小结 14](#_Toc5155)

[第3章 推荐算法选择和应用 15](#_Toc5324)

[3.1 基于用户的协同过滤算法介绍 15](#_Toc28259)

[3.1.1 User CF的实现过程 16](#_Toc13211)

[3.2 基于项的协同过滤算法介绍 17](#_Toc32751)

[3.2.1 Item CF实现过程 17](#_Toc20781)

[3.3 User CF 和 Item CF 的对比 19](#_Toc23605)

[3.4 本章小结 19](#_Toc6789)

[第4章 协同过滤推荐算法实验与应用 20](#_Toc28240)

[4.1 协同过滤推荐算法实验设计 20](#_Toc32746)

[4.1.1 实验1：基于杰卡德相似度算法的Item CF实验 20](#_Toc14493)

[4.1.2 实验2：基于余弦相似度算法的Item CF实验 21](#_Toc18607)

[4.1.3 实验3：基于杰卡德相似度算法的User CF 22](#_Toc8674)

[4.2 协同过滤推荐算法实验结果 23](#_Toc4672)

[4.2.1 协同过滤相似性算法分析： 23](#_Toc17585)

[4.2.2 推荐算法实验评测 24](#_Toc31856)

[4.3 本章小结 25](#_Toc6898)

[第5章 个性化视频推荐网站的设计 26](#_Toc5469)

[5.1 网站技术分析 26](#_Toc28348)

[5.1.2 服务端技术介绍 26](#_Toc26988)

[5.1.2 数据库技术 26](#_Toc736)

[5.1.3 网页前端展示技术 26](#_Toc42)

[5.1.4 推荐系统采用技术方案 26](#_Toc14542)

[5.2 需求分析设计 27](#_Toc5387)

[5.2.1电影视频播放网站首页 27](#_Toc26311)

[5.2.2.用户个人信息管理 27](#_Toc4968)

[5.2.3.用户行为管理 27](#_Toc26743)

[5.2.4.站内搜索功能 27](#_Toc22314)

[5.2.5.推荐功能 27](#_Toc1376)

[5.3 数据库设计 28](#_Toc32766)

[5.3.1 用户数据表 28](#_Toc25551)

[5.3.2 电影数据表 29](#_Toc21264)

[5.3.3 电影播放记录表 30](#_Toc13348)

[5.3.4 缓存数据表 31](#_Toc18321)

[5.4 本章小结 33](#_Toc24505)

[第6章 个性化视频推荐网站的实现 34](#_Toc31907)

[6.1 环境搭建 34](#_Toc8113)

[6.2 用户网站功能实现 34](#_Toc2902)

[5.2.1电影视频播放网站网页设计 34](#_Toc17257)

[5.2.2.用户个人信息管理 35](#_Toc32219)

[5.2.3.用户行为管理 36](#_Toc27503)

[5.2.4.站内搜索功能 37](#_Toc15652)

[5.2.5.推荐功能 37](#_Toc25892)

[6.3 后台管理功能实现 38](#_Toc4216)

[6.3.1电影信息管理 38](#_Toc29811)

[6.3.2电影资源管理： 38](#_Toc26318)

[6.3.3电影播放热度统计 38](#_Toc1095)

[6.3.4用户信息管理 39](#_Toc8176)

[6.4 网站前端首页功能实现 39](#_Toc15268)

[6.5 本章小结 39](#_Toc31663)

[第7章 总结与展望 40](#_Toc9355)

[7.1论文工作总结 40](#_Toc18348)

[7.2 论文展望 40](#_Toc15132)

[参考论文 41](#_Toc26758)

[致 谢 43](#_Toc11911)

[附 录 44](#_Toc371)

第1章 绪 论

## 1.1 研究背景

近二十年，随着科技进步，互联网产业的发展，人们踏入大数据时代，互联网已经融入人们的日常生活。人们可以通过智能手机、平板电脑、个人计算机、智能手环及智能手表等网络设备接入到互联网，实现远程管理、远程监控、远程办公、移动支付、网络就医、在线学习、了解时事、信息交流及在线购物等操作，互联网将人类社会从独立的点联结成一张复杂、巨大的网。

由于互联网的网络节点繁多、拓扑结构复杂，因此信息流动复杂且迅速，首先信息来源有各大媒体、社交网络、电子商务网站、网络互娱公司、信息传媒平台、网民个人等源头，其次信息消费者有网民个体、各大数据分析公司、国家机关、行业公司等。例如亚马逊通过用户信息了解用户的个人偏好、收入水平等信息制定公司发展战略，信息数据俨然成为当前大数据时代最重要的战略资源。其次，用户经常会被各种推销消息所打扰，从推荐商品、购物、贷款、买房买车到各种诈骗、虚假信息，究其原因在于个人信息的泄露导致被垃圾数据所困扰。

如何合理、合法的利用数据信息资源，发展经济效益成为时代的主题。从用户个人角度，高效地查找所需信息可以节约个人时间，提高工作、生活效率；从公司发展角度，高效的信息过滤可以带给公司在同行业的竞争优势，提高公司商品的销售额。综上所述,商品和消费者之间如何高效、准确地匹配成为公司的财富密码，推荐系统成为了大数据时代发展方向。

结合电影观影网站的行业，上述问题引申为为电影和观众之间高效、准确地匹配问题，帮助用户找到心仪的电影带来经济效益。假如用户明确自己的需求，可以借助搜索引擎来查找得到自己喜欢的电影进行付费观影，例如用户甲想看《复仇者联盟3》时，可以在网站中搜索电影名进行付费观影；但是如果用户甲需求不明确时，应该如何找到喜欢的电影，即用户想找一部自己喜欢的电影，但是不知道具体是什么，面对网站中数以万计的电影只能漫无目的的搜索，低效的搜索只会浪费用户的耐心导致用户放弃从而带来极差的用户体验，这类大部分用户都会遇到的问题叫：信息过载（Information Overload）。

为了处理信息过载问题，需要有一个根据用户需求能有效快速地为用户推荐信息的系统，它可以将网络信息过滤，将可能有用的信息分发给需要的用户，对于用户而言，可以将大海捞针变成湖里捞针，解决用户的选择困难问题，帮助用户获得良好的体验，从而获得心理上的满足，并且节省用户时间，帮助用户在有限的时间高效的完成目的；对于内容提供者来说，可以更加高效地分发内容，投其所好，赢得用户口碑，获得用户的青睐，节省系统的资源等不必要的消耗，在同行竞争者中占据先机，最终提高经济效益。这样的系统可以帮助内容提供者和内容消费者带来双赢。

## 1.2 研究目的和意义

在实际使用中，我们发现视频网站总能贴用户心意地推荐合适的电影、电视节目，那么这些视频网站（内容提供者）是如何为用户（内容消费者）推荐喜欢的电影，这个过程的背后原理是什么，视频网站如何掌握用户喜好。首先，从信息获取的角度将人类获取信息的方法分为主动搜索与被动推荐的方式：首先以线下分析：一个人想要主动知道了解信息，首先会问周围认识的人是否知道，如果知道成功获取信息，如果不知道但知道可能知道就会得到一条线索，如果什么都不知道，那这条线索就断掉，将获取到的线索集合再次搜索；被动推荐是：你的朋友猜你想知道这个，问你我有一个消息你有没有兴趣知道来主动告诉你。

再类比到线上，主动搜索和被动推荐分别对应搜索引擎和推荐系统，搜索引擎是依靠用户主动在庞大的影视资源中搜索自己所心仪的电影、电视节目，但是主动查找需要很强的目的性，否则会浪费用户大量无效时间；而推荐系统虽然节省了用户时间，但是却将问题交付给内容提供者来处理，网站需要花大量时间获取用户数据、了解用户喜好，这个过程中充满各种问题。

个性化推荐系统通过用户数据用推荐算法来帮助用户进行查找和筛选信息，为用户提供充足、有效、合适、多样、新颖的信息帮助用户排忧解难，有效的推荐系统可以帮助减少用户的搜索时间，提高用户体验，但是推荐系统需要具有尽可能快的运行、响应速度、尽可能少的资源占用，既要提高效率，又要提高速度，因此对于推荐算法的优化成为解决这项问题的重要手段。

通过建设一个小型的视频网站的过程来研究协同过滤推荐算法的原理和实现方法，利用推荐系统优化视频网站，赋予网站个性化的功能，能够为不同的用户定制不同的主页，推荐用户所喜欢的电影、电视节目。

使用推荐算法系统改进视频网站的个细化功能，实验使用不同的相似性算法时的推荐算法、实验评价性算法，了解推荐系统的运作原理。

## 1.3 研究现状

### 1.3.1 推荐算法发展

推荐算法和搜索引擎相似，但是搜索是主动的，而推荐系统根据用户的历史记录进行用户喜好推荐，用户处于一个被动的状态，推荐系统往往作为一个应用网站的子系统来进行设计，依赖网站的数据来进行推荐活动，而非单独存在。在互联网的各类网站中，推荐系统应用很广，从电商、视频、直播、新闻、媒体到广告、娱乐、游戏等都有推荐系统的影子，通过大量分析用户行为数据，给用户提供个性化服务来提高网站流量、吸引更多的用户、提高市场占有率，最终提高经济效益，个性化推荐系统应用于购物类网站、音乐类网站、电影和视频类网站、阅读类网站、社交和新闻类网站、小视频和直播类网站等。

个性化推荐算法的研究开始于上世纪90年代，随着互联网的发展开始应用于各行各业，随着行业发展需求不断的变化，对推荐系统的要求越来越高。个性化推荐系统按照内容可以分为（1）用户模块，即推荐的目标人群；（2）内容模块，即待推荐的主要内容；（3）推荐算法，将内容推荐给用户的规则，是连接用户与内容的核心桥梁，也是需要深入研究的内容。

### 1.3.2 推荐算法的研究现状

协同过滤推荐算法（Collaborative Filtering,CF）目前应用较多，是一种成熟的推荐算法，目前在工业生产中被广泛推荐。其基本思想是根据用户历史行为数据推荐用户喜好。CF算法主要有三个流程：（1）数据处理，对于用户历史行为数据的数据处理；（2）相似性处理，根据历史数据确定相似性矩阵，对用户或者内容进行相似性计算；（3）相似性推荐，根据相似性矩阵为用户进行推荐。根据推荐系统要处理的问题可以分为：（1）评分预测；（2）Top N 推荐。

在推荐系统的发展中长期存在着一些问题：冷启动问题；数据稀疏问题；用户喜好偏移问题。冷启动问题即新用户、新项目没有历史数据的情况下如何加入到推荐系统中来；数据稀疏的问题是指海量的用户和项的情况下，用户数据稀少，导致用户相似性矩阵或物品相似性矩阵稀疏；用户兴趣漂移是指用户的兴趣变化，因为人是具有思维的个人，具有相当大的不确定性、偶然性有可能某一天用户的兴趣突然就变化180度，那么对于原先的相似性矩阵就无法适用了。

### 1.3.3 国内外推荐算法应用现状

个性化推荐系统在电影和视频网站中具有重要的应用地位，因为电影视频资源的数量庞大、不可遍历，导致用户对于海量信息的搜索存在巨大的困难。

腾讯视频网站的推荐系统，腾讯视频网站可以根据电影之间的相似性进行归类，根据用户播放记录帮助用户筛选用户可能感兴趣的电影；也可以根据用户好友的之间的联系，利用QQ、微信中的好友数据资源，来推荐用户好友圈子中受欢迎的电影节目。

QQ音乐平台的推荐，主要依据用户的历史记录和好友圈的历史记录来分析，通过比对所有人音乐的喜好来计算每首音乐之间的相似性来，通过用户的听歌记录来生产用户的私人歌单，通过所有用户的播放记录找到两个听歌风格相似的用户来互相推荐歌曲。

阅文网站的小说推荐，当用户浏览一本小说时，下面会给出推荐列表，从小说的相似性出发找到同类型的小说加入到推荐列表推荐给用户。

电商应用方面以亚马逊举例推荐系统在电商的发展主要得益于亚马逊，现在电商平台中的推荐系统就成了标配，通过在数以亿计的购买记录数据中提取用户喜好信息，用数据来刻画出来用户的喜好，对相似同类人、商品进行归类，掌握每一个人的性格、喜好等，通过用户喜好进行用户推荐。

在推荐列表应用中包含着商品的显性信息和隐形信息以及相关属性，其他用户对商品的评分，综合评价出两个商品的相关性。显性的信息是两个商品明面上的相似处，例如两本书之间的相似，而隐形的相似性是从用户的购买记录中发现的相似规律，例如笔记本和笔会经常被用户同时购买。在亚马逊的个性化推荐系统中，最重要的算法是一种基于物品的推荐算法，给用户推荐用户喜欢物品的相似物品，另一种推荐算法是基于用户的推荐算法，亚马逊是利用Facebook上的好友信息，判定好友之间的相似度来推荐商品。使用推荐系统的直接好处是给亚马逊公司带来直接的经济效益，据统计，亚马逊百分之三十左右的商品销售来自于推荐系统。

## 1.4 本文安排

第1章，绪论，介绍本课题研究背景、研究意义和目的，研究现状。

第2章，推荐算法、相似度算法相关介绍，推荐算法评价方案。

第3章，推荐算法优势对比、选择以及实现过程介绍。

第4章，协同过滤推荐算法的实验

第5章，个性化视频网站需求分析、功能设计及数据库设计规划

第6章，个性化视频网站功能实现

第7章，总结与展望，优化与修改

1. 相关技术总览介绍

当前主流的推荐系统主要分为：基于内容的推荐系统（Content Based, CB）、协同过滤推荐系统（Collaborative Filtering,CF）、混合推荐系统，其中混合推荐系统是一种综合性推荐系统，是融汇各种推荐算法优点来设计，因为每种推荐算法都有其应用范围、限制，在不同的条件下，推荐效果各有差异，因此开发混合推荐系统成为当前的主流的研究方向。

## 2.1推荐算法总览

### 3.1.1基于内容的推荐

CB推荐（Content Based，CB)，主要用于推荐文本类项目，采用浅层模型分析用户历史记录，提取核心内容词汇，根据物品的便签属性、内容关键词等用户特有的兴趣相关的词汇，推荐有相似标签的物品（文本、商品、电影等）。例如：一个人喜欢看《夏洛特烦恼》那么我可以给他推荐同为喜剧的《人在囧途》、《飞驰人生》等，他们的内容标签可以为：类型喜剧，主演：沈腾、尹正，例如一个人购买了数据结构书籍，可以根据内容推荐其相关算法书籍。不能根据评分或名称，需要根据物品本质进行推荐。 比较最直观，推荐理由也很充分，但需要做好新物品的归类，内容标签的标定，并且推荐精度较差，结果惊喜度不足，你所推荐的正式用户所想的。

### 3.1.2 基于内存的协同过滤

协同过滤(Collaborative Filtering, CF)是当前推荐算法中比较常用和经典的推荐算法，种类繁多，目前在生产环节又很多的应用，基本原理就是根据用户的历史行为数据归类出相似性，发现用户、物品或者内容间的相关性进行推荐，如果用户喜欢A，那么就像和B相似度B推荐给用户。基于内存的协同推荐算法的计算过程全放在内存之中 ，虽然实现简单、直观，但是不能为用户实时推荐，每次有新内容加入时需要重新计算，难以应对高并发的需求。

### 3.1.4 基于模型的协同过滤

基于模型的协同过滤算法（Model-based Collaborative Filtering），通过机器学习的方法预测用户对商品的喜好程度，主要的步骤有线下的模型训练、线上的推荐预测，利用数据集训练出较好的推荐算法模型，预测新样本时可以通过从新内容的特征参数输入到推荐算法模型中得到预测结果。常规的应用在推荐中的机器学习方法包括关联分析、聚类算法、回归算法、分类算法、矩阵分解、神经网络、隐语义模型、图模型等，随着神经网络的研究和发展，基于神经网络的推荐算法也日渐壮大。适用范围广大，可以应对数据稀疏的场景，能很好地发现用户潜在偏好。

### 3.1.5 混合推荐算法

在上文都是单一的推荐算法，每个算法都有各自的有点和缺点，在生产中为了应对复杂环境需要取长补短，因此产生混合推荐算法，混合推荐算法不是一种具体的算法，而是多种推荐算法结合得到的新的推荐算法，虽然提高了算法的复杂度，但是可以利用各种算法的互补提高推荐系统健壮性。

## 2.2 相似度算法总览

推荐算法中具有一个核心的问题：相似度的判断，如何衡量两个项之间的相似性，需要将抽象的关系转化为一个具象的数据，通过建立项的数学模型转化为向量，通过向量之间的相似性来比较，常见的相似性算法如下。

### 3.2.1 欧几里得距离

欧氏距离（Eucledian Distance）是常用的距离计算公式，是从平面直接坐标系两点间距离公式推广产生，其含义为n维空间坐标系中两个点的绝对距离，通过构建n维向量可以简单地计算距离，是一种非常方便的计算方式。

但其缺点是容易受到尺度影响，即计算的距离会因为特征维度的单位导致结果，例如在二维空间坐标系中，x为身高，单位是厘米（cm），y为体重，单位是千克（kg），这两个单位不同的指标计算的结果会发生偏移，导致结果没有参考价值。因此在使用欧式距离度量之前，需要对数据进行归一化处理，当然在单位尺度不同的条件下使用距离公式都需要特别注意尺度问题。

### 3.2.2 曼哈顿距离

式（1）

曼哈顿距离（Manhattan Distance）是简化欧式距离而产生，由于欧式距离中存在大量平方、根号操作计算时间长，因此将欧式距离进行简化。曼哈顿距离又称街区距离、国际象棋距离。

式（2）

### 3.2.3 切比雪夫距离

切比雪夫距离（Chebyshev Distance）是在空间向量中，两点之间的距离坐标差值绝对值最大值的距离。

式（3）

### 3.2.4 闵可夫斯基距离

闵可夫斯基距离（Minkowski distance）是一种距离的定义，是对多种距离度量公式的概括性表述，例如：

当p等于无穷大时,作为切比雪夫距离

当p=2时，作为欧式距离

当p=1时，作为曼哈顿距离

式（4）

### 3.2.5 余弦相似度

余弦相似度（Cosine Similarity）通过计算两个向量的夹角余弦值来评估他们的相似度，将向量分布到直角坐标系（n维向量分布到n维空间直角坐标系中），绘制从零点出发的向量空间，求两个向量之间夹角余弦值，表示这两个向量的相似性，当夹角越小，余弦值越接近于1，它们的方向越吻合，则越相似，反之则不相似。

式（5）

### 3.2.6 杰卡德相似度

杰卡德相似度（Jaccard Similarity），是在两个集合A、B中，利用A、B交集元素的个数比上A、B并集中元素的个数来确定集合的相似度，该值称为这两个集合的杰卡德系数。

式（5）

## 2.3 推荐系统评测方案

如何评价一个推荐系统的优劣，什么样的推荐系统是我们所需要的，目前还没有一个统一的标准，常规的推荐系统评价方案有：离线实验（offline experiment）、用户调查（user study）和在线实验（online experiment）。

### 3.3.1 离线实验

1. 离线实验通常用于开发阶段的测试，使用离线数据集评测，具体过程如下：

1. 获得用户行为数据

2. 将数据集按照一定的规律分成训练集和测试集

3. 利用训练集训练推荐模型

4. 利用离线评测指标在测试集上进行预测结果的评价

5. 根据评测结果对于推荐算法进行改进

2. 离线实验的优点是：

1. 不需要用户参与、时间经济成本低

2. 利用离线数据验证、时间周期短、效率高

3. 可以无人值守，调试修改算法灵活

3. 离线实验的缺点是：

1. 受到数据集合影响大，例如数据集稀疏、片面等局限性

2. 对于评价结果缺乏主观性，无法考虑到人的主观思想变化，缺乏灵活

3. 对于用户的喜好很难产生一个具体的数值评价标准

4. 对于用户的变化无法做到及时响应，例如用户的兴趣偏移

### 3.3.2 用户调查

用户调查需要真实的用户在待测推荐系统上完成任务，并且观察和记录用户的行为以及回答一些问题，分析用户的行为和答案，了解测试系统的性能。

用户调查的优点是：

1. 可以获得用户主观感受的指标，对于用户的主管想法进行记录和考究
2. 测评范围小，造成风险的影响范围较小，风险易控制。

用户调查的缺点是：

1. 需要招募测试用户，时间经济方面消耗较大
2. 无法组织完全覆盖的测试用户，统计结果片面

### 3.3.3 在线实验

通过离线实验和用户调查测评之后，可以将系统上线做AB测试，即将新的推荐系统与旧的推荐系统共同投入使用进行测试，通过两个推荐系统的对比进行测评，并且减少风险。

在线实验通过一定的规则将用户随机分成几组，对不同组的用户采用不同的推荐算法，然后通过统计不同组的评测指标，比较不同算法之间的优劣，其的核心思想是:

1. 控制变量（推荐算法的选择）进行多方案共同运行
2. 比较各个方案测试结果
3. 以某种规则进行修改优化或者是淘汰

在线实验的优点是：

1. 评价结果具有很强的参考性
2. 在实际环境出发，能客观、准确地反应推荐系统优劣

在线实验的缺点时：

1. 评价过程比较复杂，对于时间、人力、物力耗资巨大
2. 具有不可控的风险，有造成潜在经济效益损失的风险

## 2.4推荐算法评测指标

推荐系统本质上属于一种分类模型，因此可以采用分类模型评价的指标对推荐系统的结果进行评测。

### 2.4.1混淆矩阵及其相关指标

混淆矩阵也称误差矩阵（Confusion Matrix），是评判模型结果的指标，属于评估模型的一种，多使用在判断分类模型的优劣。在分类实验中将真实结果、预测结果的真假共四种结果分布到2 x 2的表格中进行分析，如表2-4-1-1所示。

|  |  |  |
| --- | --- | --- |
| 真实结果 | 预测结果 | |
|  | 真 | 假 |
| 真 | True Positive | True Negative |
| 假 | False Positive | False Negative |

表2-4-1-1 混淆矩阵

混淆矩阵指标介绍：

TP（True Positive）：将真值预测为真，真正类，正确预测

TN（True Negative）: 将真值预测为假，伪假类，错误预测

FP（False Positive）: 将假值预测为真，伪正类，错误预测

FN（False Negative）：将假值预测为假，真假类，正确预测

根据上述四个指标得到二级指标对预测结果评价：

1. 准确率（Accuracy）：预测正确的数量（真正类+真假类）与所有预测数量之比，如式（1）

式（1）

1. 精确率（Precision）：将真类预测正确的数量（真正类）与所有预测为正确的数量（真正类、伪正类）之比，如式（2）

式（2）

1. 召回率（Recall）：将真类预测正确的数量（真正类）与所有真类的数量（真正类、伪假类）之比，如式（3）

式（3）

### 2.4.2 ROC曲线和AUC面积

ROC（Receiver Operating Characteristic）曲线和AUC（Area Under the Curve）面积常常用来评价一个二值分类器的优劣。

1. ROC曲线是以FPR做横坐标、TPR做纵坐标绘制出的曲线。其中FPR为假正例率，TPR真正例率，定义如下：

FPR：真正类的数量在所有预测为正类的数量的比例

TPR：伪正类的数量在所有预测为假类的数量的比例

式（4） 式（5）

2. AUC面积是ROC曲线向下、向右与x、y轴围成的面积

### 2.4.3 F分数

F分数（F score），可以理解为精确率和召回率在不同权重下的调和结果，是为了解决精确率和召回率不能同时满足的问题做出的取舍，当精确率重要时，提高精确率权重，反之提高召回率权重。如式（6）

式（6）

当β=1时，称为F1 score，此时精确率和召回率权重相同，即精确率和召回率一样重要，式（6）可以转换为式（7）

式（7）

### 2.4.4 评分预测评价指标

对于推荐结果会有一个用户打分的功能，收集用户对推荐结果的反馈分数和推荐系统预测的分数进行一个误差计算来评估推荐系统效果的优劣。具体的评分误差评价方法有均方误差（MSE）、均方根误差（RMSE）等，误差越小，说明系统推荐准确性更高

1. 均方误差MSE是真实值和预测值的差值平方求和取平均值

式（8）

1. 均方根误差RMSE是均方误差的开平方

式（9）

## 2.5 本章小结

本章节主要介绍推荐算法的相关基础知识，推荐算法的类型分类、推荐算法原理，推荐算法的数学基础和推荐算法的核心相似性算法原理，以及推荐算法最后的评价标准。

第3章 推荐算法选择和应用

基于内存的协同过滤（Memory-Based Collaborative Filtering）可以细分为基于用户的协同过滤推荐（User-based Collaborative Filtering）、基于项的协同过滤推荐（Item-based Collaborative Filtering），这两种方法推荐方式不同之处在于建立的相似性矩阵、和如何相似性矩阵的方法不同。

## 3.1 基于用户的协同过滤算法介绍

User CF基本思想是根据用户历史行为记录发现兴趣偏好相似的邻居用户群，根据邻居用户群体的行为记录的并集来为用户推荐，如图3-1-1.1，甲和乙观影记录相似都看过电影1、电影5可得甲（电影1、电影3、电影5），乙（电影1、电影4，电影5），可以得出甲、乙的偏好相似，甲乙这类人喜好观看电影1、电影3、电影4、电影5，因此在甲、乙之间互相推荐得效率更高，因此给甲推荐电影4，给乙推荐电影3。



图3-1-1.1

User CF接近于用户兴趣群体推荐，先发现兴趣圈再交流兴趣，比较精准，适用物品多且杂的场景，具有跨领域推荐新信息的能力，可以发现用户潜在但自己尚未察觉的潜在兴趣，惊喜度高，能够推荐艺术品、音乐、电影和一些难以进行内容分析的产品，具有跨领域能力。

### 3.1.1 User CF的实现过程

1. 根据用户与系统的交互行为，收集用户评分数据表，如表3-1-1.2，在用户评分数据表基础上建立用户-项评分矩阵，如表3-1-1.3。

|  |  |  |
| --- | --- | --- |
| userId | movieId | rating |
| 1 | 1 | 4 |
| 1 | 3 | 4 |
| 1 | 6 | 4 |
| 2 | 3 | 5 |
| 2 | 5 | 5 |
| 2 | 7 | 3 |
| 2 | 10 | 5 |
| 3 | 1 | 4 |
| 3 | 5 | 5 |
| 3 | 3 | 5 |
| 4 | 3 | 5 |
| 4 | 2 | 5 |
| 4 | 5 | 3 |

表3-1-1.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| userId | movieId | | | | | | |
|  | 1 | 3 | 6 | 5 | 7 | 10 | 2 |
| 1 | 4 | 4 | 4 |  |  |  |  |
| 2 |  | 5 |  | 5 | 3 | 5 |  |
| 3 | 4 | 5 |  | 5 |  |  |  |
| 4 |  | 5 |  | 3 |  |  | 5 |

表3-1-1.3

2. 根据上述用户-电影评分矩阵中的用户对电影的评分数据，在结合相似性计算方式，确定偏好用户之间的相似性集合。如表3-1-1.4，是利用余弦相似度算法计算的用户之间相似度。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| userId | 1 | 2 | 3 | 4 |
| 1 |  | 0.33502969713 | 0.64333315709 | 0.39589732744 |
| 2 | 0.33502969713 |  | 0.68931234948 | 0.59083915670 |
| 3 | 0.64333315709 | 0.68931234948 |  | 0.69333333333 |
| 4 | 0.39589732744 | 0.59083915670 | 0.69333333333 |  |

表3-1-1.4

3. 根据用户的相似度度矩阵，可以得出目标用户的相似邻居，由邻居用户的行为记录中筛选出推荐电影给目标用户。

## 3.2 基于项的协同过滤算法介绍

Item CF 主要考虑项与项之间的联系，通过确定相似的项目集合，然后根据用户历史记录寻找相似的项进行推荐，原理如图3-1-2.1，电影1、电影5都有相同的关注甲丙，所以电影1、电影5相似，用户乙看过电影5，所以推荐给他和电影5相似的电影1。

Item CF则更接近个性化推荐，适用于物品较少的场合（便于建立物品相似矩阵），应用场景有豆瓣的豆瓣猜、淘宝的猜你喜欢、百度的猜你想搜、各大音乐播放器的猜你想听及各大视频网站的猜你想看等，同时Item CF还可以给推荐提可信的推荐理由，例如：淘宝上最近购买物品A的人也购买了物品B，亚马逊的收藏物品A的人在搜索物品B。因为同时购买物品A、B的行为比较多，因此判定A、B间具有很大的相似性，推荐理由充分。

基于项目的推荐算法尤其在电商行业应用广泛，通过分析所有用户对物品的偏好找到物品之间的相似性联系，构建物品相似性矩阵，借助用户的物品行为记录信息来为用户推荐相似物品，推荐精度高，倾向于推荐物品比较稳定的场景，能较快得出在线结果，但是对于物品量多、更新快的场景效果不足，并且在物品数据稀疏，难以构建有效的物品相似度矩阵时，效果较差。



图3-1-2.1

### 3.2.1 Item CF实现过程

1. 首先根据用户的历史评分数据表，如表3-1-2.2，建立项-用户评分矩阵，如表3-1-2.3。

|  |  |  |
| --- | --- | --- |
| userId | movieId | rating |
| 1 | 1 | 4 |
| 1 | 3 | 4 |
| 1 | 6 | 4 |
| 2 | 3 | 5 |
| 2 | 5 | 5 |
| 2 | 7 | 3 |
| 2 | 10 | 5 |
| 3 | 1 | 4 |
| 3 | 5 | 5 |
| 3 | 3 | 5 |
| 4 | 3 | 5 |
| 4 | 2 | 5 |
| 4 | 5 | 3 |

表3-1-2.2 历史评分数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MovieId | userId | | | |
|  | 1 | 2 | 3 | 4 |
| 1 | 4 |  | 4 |  |
| 2 |  |  |  | 5 |
| 3 | 4 | 5 | 5 | 5 |
| 5 |  | 5 | 5 | 3 |
| 6 | 4 |  |  |  |
| 7 |  | 3 |  |  |
| 10 |  | 5 |  |  |

表3-1-2.3 项-用户评分矩阵

1. 根据上述用户-电影评分矩阵中的用户对电影的评分数据，在结合相似性计算方式，物品-物品相似矩阵，并且不同的相似度计算方法最终会得到不同的结果，如表3-1-2.4 项相似性矩阵。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MovieId | 1 | 2 | 3 | 5 | 6 | 7 | 10 |
| 1 | 1 | 0 | 0.6671 | 0.4603 | 0.7071 | 0 | 0 |
| 2 | 0 | 1 | 0.5241 | 0.3906 | 0 | 0 | 0 |
| 3 | 0.6671 | 0.5241 | 1 | 0.8871 | 0.4193 | 0.5241 | 0.5241 |
| 5 | 0.4603 | 0.3906 | 0.8871 | 1 | 0 | 0.6509 | 0.6509 |
| 6 | 0.7071 | 0 | 0.4193 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0.5241 | 0.6509 | 0 | 1 | 1 |
| 10 | 0 | 0 | 0.5241 | 0.6509 | 0 | 1 | 1 |

表3-1-2.4 项相似性矩阵

1. 根据物品相似度矩阵，结合用户电影矩阵，找到和用户已观看电影相似的电影列表，生成推荐电影Top N列表。

## 3.3 User CF 和 Item CF 的对比

通过上文User CF 和 Item CF 的流程可知，User CF、Item CF实现过程相似，但得到的相似性矩阵不同，如表3-3.1。

1.从计算主体上来看，User CF 主要计算用户之间的相似性，最终得到用户-用户相似性矩阵，而Item CF 主要计算项目之间的相似性，最终得到物品-物品相似性矩阵。

2.从应用领域上来看User CF 倾向于找到用户临近的聚类，为用户找到共同爱好群，常与社交功能结合，相当于是一个同好交流群；Item CF 则是为项目寻找聚类，对于项目的较为敏感，常用于电商平台等。

3.从实时性上看，User CF 是根据用户的近邻用户来推荐，对于用户行为发生变化的场景，推荐结果变化不够迅速，适合离线推荐；Item CF 是根据用户历史行为进行推荐，可以增加用户新记录的权重，加快对用户新行为记录的响应，当用户行为发生变化时，推荐结果可以快速变化。

|  |  |  |
| --- | --- | --- |
|  | User CF | Item CF |
| 性能 | 运算量与用户数量有关 | 运算量与物品数量有关 |
| 领域 | 时效性高，个性化低 | 物品丰富，个性化要求 |
| 实时性 | 不足 | 用户行为变化响应较快 |
| 冷启动 | 新用户不友好 | 新用户、新物品可以快速加入推荐 |
| 推荐理由 | 按照相似用户群体推荐 | 根据用户看过推荐 |
| 跨领域性 | 跨领域推荐 | 同类物品推荐 |
| 核心矩阵 | 用户相似性矩阵 | 物品相似性矩阵 |

表3-3.1

## 3.4 本章小结

本章介绍基于用户的协同过滤算法和基于项目的协同过滤算法的实现过程，以及两者之间的差异不同之处。

第4章 协同过滤推荐算法实验与应用

## 4.1 协同过滤推荐算法实验设计

本论文实验采用环境如表4-1.1，

|  |  |
| --- | --- |
|  | 环境配置 |
| 操作系统 | Windows 10 家庭版64位 |
| 理器 | Intel (R)Core(TM) i5-7300HQ |
| 磁盘 | 300 G固态 |
| 内存 | 16 G |
| 编程语言 | Python 3.7 |
| 数据集 | MovieLens 100k |

表4-1.1

MovieLens数据集由GroupLens实验室采集上传的电影评分数据信息表，采用ml-latest-small数据集，数据集中包括600多个用户9000部电影，10万条评分信息，实验过程中使用随机方法划分训练、测试数据。

Python作为当前热门的编程语言，代码语义清晰简短、方便编写人员理解配置，可以移植性强、能跨平台使用，具有强大的数据处理能力，有着非常方便的技术支持，例如Numpy开源科学计算库、pandas数据分析库等。

NumPy是基于Python 语言开发的一个开源数值计算扩展程序库，支持大量的维度数组与矩阵运算，是一个高效的数学计算库。

### 4.1.1 实验1：基于杰卡德相似度算法的Item CF实验

实验过程：

1. 首先读入用户的历史行为数据表（用户行为数据表），
2. 建立如下所样式的movie-user矩阵

|  |  |
| --- | --- |
| movie | set() |
| movie1 | user1,user2,user3,user4 |
| movie2 | user2,use3,user4,user5,user6 |
| movie3 | user5,user6,user9 |
| movie4 | user3,user6,user7 |

1. 利用电影之间的观影数据的交集比并集算出电影的相似性

核心程序：

def jacard(u1,u2):

return len(set.intersection(u1, u2)) / len(set.union(u1,u2))

1. 根据物品相似度矩阵，结合用户电影矩阵，找到和用户已观看电影相似的电影列表，生成推荐电影Top N列表

def recommend(user, K, N, trainSet, movie\_sim\_movie):

rank = {}

watched\_movies = trainSet[user]

for movie, rating in watched\_movies.items():

for related\_movie, w in sorted(movie\_sim\_movie[movie].items(), key=itemgetter(1), reverse=True)[:K]:

if related\_movie in watched\_movies:

continue

rank.setdefault(related\_movie, 0)

rank[related\_movie] = w \* float(rating)

return sorted(rank.items(), key=itemgetter(1), reverse=True)[:N]

由于杰卡德相似度无法体现用户评分数据，因此利用相似度和用户观影的电影评分估计出推荐电影评分。

### 4.1.2 实验2：基于余弦相似度算法的Item CF实验

实验可以分为以下过程：（仅描述不同步骤)

不同项：建立movie-user矩阵时，需要对空余值补0，

|  |  |
| --- | --- |
| movie | List() |
| movie1 | user1,user2,user3,user4,0,0,0,0,0 |
| movie2 | 0,user2,use3,user4,user5,user6,0,0,0 |
| movie3 | 0,0,0,0,user5,user6,0,0,user9 |
| movie4 | 0,0,user3,,0,0,user6,user7,0,0 |

不同项：使用余弦相似度公式计算电影之间相似度

核心程序：

def cosine(movie1,movie2):

x = to\_array(movie1)

y = to\_array(movie2)

xy = np.sum(x\*y)

x = np.linalg.norm(x,2)

y = np.linalg.norm(y,2)

return xy / (x\*y)

### 4.1.3 实验3：基于杰卡德相似度算法的User CF

相似度计算过程：

（1）首先读入用户的历史行为数据表（用户行为数据表）

（2）建立user\_movie矩阵，形如：

|  |  |
| --- | --- |
| user | set() |
| user1 | movie1,movie2 ······ |

核心代码

def calc\_user\_sim(trainSet):

user\_sim\_matrix = {}

for u1, m1 in user\_movie.items():

for u2, m2 in user\_movie.items():

if u1 == u2:

continue

else:

user\_sim\_matrix.setdefault(u1, {})

user\_sim\_matrix[u1].setdefault(u2, 0)

user\_sim\_matrix[u1][u2] = len(set.intersection(m1, m2)) / len(set.union(m1,m2))

return user\_sim\_matrix, user\_movie, movie\_count

（3）根据上述用户间相似性矩阵中的用户对电影的评分数据，在结合相似性计算待推荐电影的评分数据，生成推荐电影Top N列表。

推荐代码

def recommend(user, K, N, trainSet, user\_sim\_matrix):

rank = {}

watched\_movies = trainSet[user]

for v, wuv in sorted(user\_sim\_matrix[user].items(),key=itemgetter(1), reverse=True)[0:K]:

for movie,rating in trainSet[v].items():

if movie in watched\_movies:

continue

rank.setdefault(movie, 0)

rank[movie] = wuv \* rating

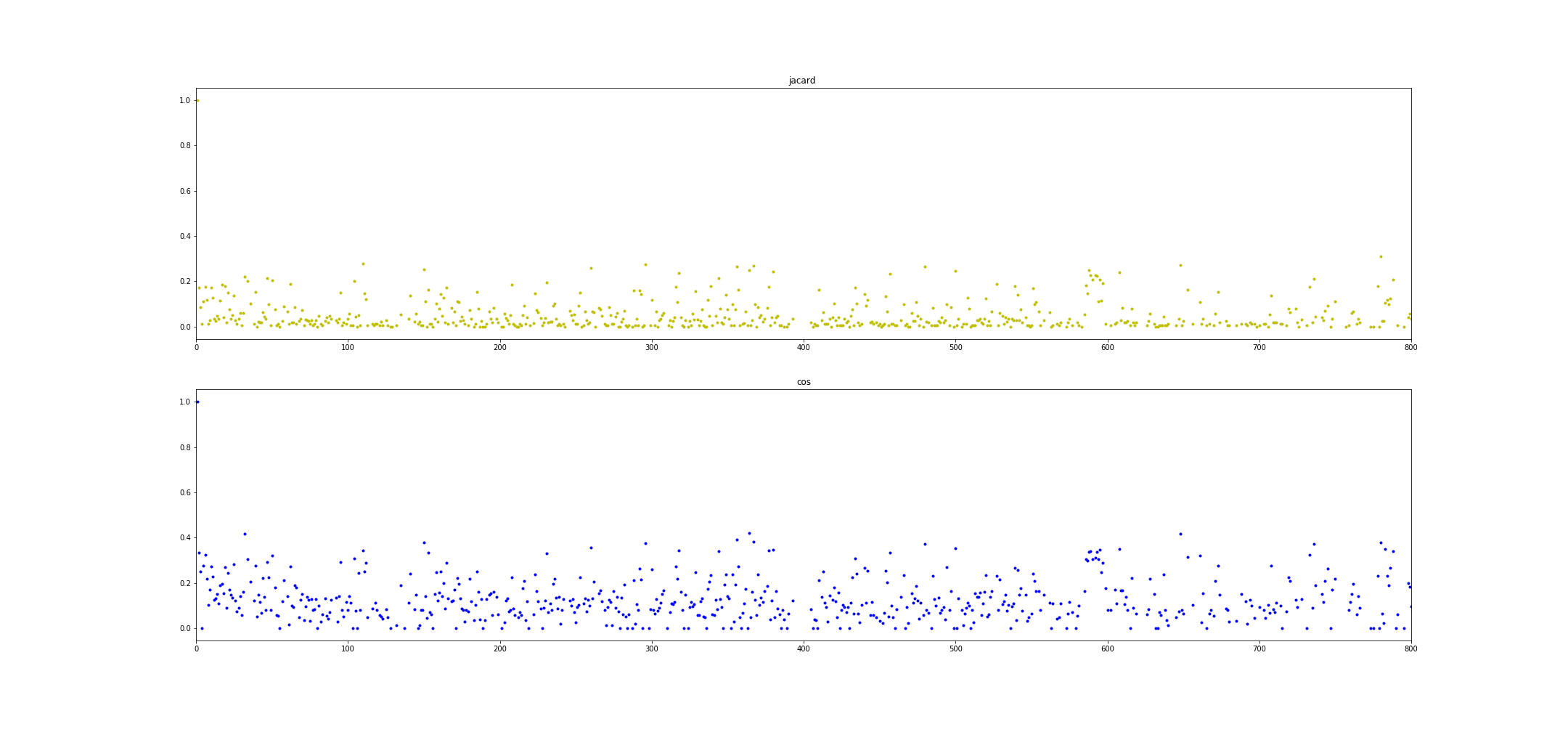
return sorted(rank.items(), key=itemgetter(1), reverse=True)[0:N]

基于用户的协同过滤推荐系统需要先找到用户相似的用户群里，并且利用相似用户的观影记录来进行推荐

## 4.2 协同过滤推荐算法实验结果

### 4.2.1 协同过滤相似性算法分析：

相似性算法结果集合：截取800部电影。



由图可以观察：余弦相似度、杰卡德相似度算法的峰谷接近，相似度趋势相似；在余弦相似度的结果中，相似度结果之间区分度比较高，结果散步在0到0.4之间，在杰卡德的算法中，结果散布0到0.2之间。

在基于项目的三种相似度算法计算相似性的运行时间：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 基于项的杰卡德相似度 | 基于项的余弦相似度 | 基于用户的杰卡德 |
| 开始时间 | 21:12:40.193205 | 12:47:20.921754 | 08:24:47.405325 |
| 结束时间 | 21:15:00.385559 | 13:14:54.359994 | 08:24:51.146901 |
| 用时 | 00:02:20.192354 | 00:27:33.438240 | 00:00:03.741576 |

从代码中分析，杰卡德相似性主要依靠计数，余弦相似性需要进行余弦值的计算。在基于项的余弦相似度算法中，对于9000条电影，余弦相似度的用时远高于杰卡德相似度算法；在使用杰卡德算法中，基于项目的杰卡德算法用时高于基于用户的杰卡德算法，主要区别在于电影数量级在9000，用户数量级为600左右，由此可可见，在物品多，用户少的情况下使用基于用户的协同过滤推荐效率较高；在追求速度的基础下，杰卡德相似度可以满足基本区分要求。

### 4.2.2 推荐算法实验评测

杰卡德相似算法，基于项的推荐结果

|  |  |
| --- | --- |
| [(166, 3.333333333333333), (2607, 3.333333333333333), | |
| (2769, 3.333333333333333), (3078, 3.333333333333333), | |
| (3747, 3.333333333333333), (3862, 3.333333333333333), | |
| (5347, 3.333333333333333), (5423, 3.333333333333333), | |
| (5425, 3.333333333333333), (5486, 3.333333333333333), | |
| (6335, 3.333333333333333), (7354, 3.333333333333333), | |
| (7386, 2.5), (118, 2.5), | |
| (521, 2.5), (835, 2.5), | |
| (1040, 2.5), (2095, 2.5), | |
| (2425, 2.5), (2552, 2.5)] | |
| 开始时间 | 43:37.7 |
| 结束时间 | 43:38.0 |
| 用时 | 00:00.4 |
| MAE | 1.608143591 |
| RMSE | 2.778067166 |
| 精确率 | 0.1 |
| 召回率 | 0.039215686 |

余弦相似算法，基于项的推荐结果

|  |  |
| --- | --- |
| [(1089, 78.82580696217214), (1196, 61.282928846791435), | |
| (1210, 57.77731956398587), (1527, 57.641792240098006), | |
| (2916, 50.21300416057626), (1240, 49.40658980557108), | |
| (2918, 48.09092229144081), (2683, 41.613716850034905), | |
| (296, 41.16519846085778), (1682, 40.73998296885274), | |
| (2858, 39.9216220944197), (858, 39.22801247072661), | |
| (2997, 36.45176676147006), (589, 36.39426872679257), | |
| (1097, 36.03207582139306), (1036, 34.97743872861351), | |
| (3033, 34.08231204279323), (1580, 31.63422636389324), | |
| (1129, 30.260999081947112), (2959, 28.552920447178778)] | |
| 开始时间 | 14:41.2 |
| 结束时间 | 14:41.8 |
| 用时 | 00:00.6 |
| MAE | 1.360856857 |
| RMSE | 1.851931385 |
| 精确率 | 0.05 |
| 召回率 | 0.01818 |

杰卡德相似算法，基于用户的推荐结果

|  |  |
| --- | --- |
| [(199, 0.13819095477386933), (242, 0.13819095477386933), | |
| (627, 0.13819095477386933), (695, 0.13819095477386933), | |
| (896, 0.13819095477386933), (1175, 0.13819095477386933), | |
| (1619, 0.13819095477386933), (1900, 0.13819095477386933), | |
| (1916, 0.13819095477386933), (2275, 0.13819095477386933), | |
| (2333, 0.13819095477386933), (2568, 0.13819095477386933), | |
| (2574, 0.13819095477386933), (3266, 0.13819095477386933), | |
| (3484, 0.13819095477386933), (3909, 0.13819095477386933), | |
| (4278, 0.13819095477386933), (4298, 0.13819095477386933), | |
| (4562, 0.13819095477386933), (4901, 0.13819095477386933)] | |
| 开始时间 | 12:24.8 |
| 结束时间 | 12:24.8 |
| 用时 | 00:00.0 |
| MAE | 0.617993205 |
| RMSE | 0.381915601 |
| 精确率 | 0.05 |
| 召回率 | 0.01785 |

从结果中分析，结果的精确率、召回率较低，查找原因精确率低原因在于相似性矩阵相似度较低，由于训练数据不足，相似性结果普遍偏低，由于采用topN推荐，对于召回率有一定影响。

## 4.3 本章小结

本章验证了协同过滤推荐算法基于项和用户的实验，以及两种相似性算法的相似性矩阵结果，由实验图标可知，不同的相似性算法得到的相似性程度近似相同，因此在实验中确定使用基于杰卡德的协同过滤算法作为网站的支撑算法。

第5章 个性化视频推荐网站的设计

## 5.1 网站技术分析

### 5.1.2 服务端技术介绍

Flask是目前比较流行的基于Python的Web框架，是用Python来实现的。其显著的特点是：“微”框架。表示Flask是一个核心的简单、方便入手、同时又易于扩展的轻量级框架。默认的Flask是不包含诸如数据库支持、表单验证，或者是其他任何功能。但是易于拓展的Flask框架支持使用扩展方式来给应用添加其他功能。这些扩展包括数据库集成、表单验证、文件上传处理、其他各种各样的开放认证技术等功能，开发者可以根据自己的需求选择自己的功能，因此Flask在Web开发方面变得非常流行,本论文中推荐算法采用Python实现，因此采用作为Flask开发视频网站的技术支持。

### 5.1.2 数据库技术

MongoDB是一个基于分布式文件存储的文档型数据库，介于关系型和非关系型之间的模式，是NoSql数据库中最热门的一种，基于C++语言开发的高性能、易部署，作为NoSql型数据库，旨在为大规模web应用部署服务，MongoDB存储模式自由（可以存储不同结构的文件，对于不规则的数据存储有较好的支持），存储高效，文件以BSON格式存储（Json格式的拓展）和Python有较好的适配，适合作为本论文中网站的后台数据库。

Redis 是一个开源的高性能 Key Value 数据库，Redis将数据存储在内存中，所以读写速度非常快，并且支持数据持久化，因此Redis被广泛应用在缓存方向，可以作为分布式锁来使用，本论文中，利用Redis作为缓存数据库。

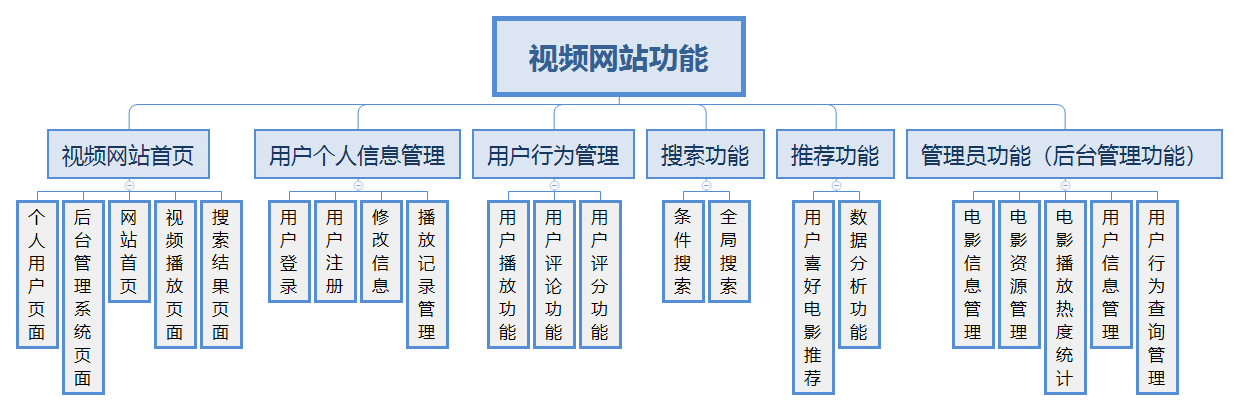
### 5.1.3 网页前端展示技术

视图展示功能使用，JinJa2 和bootstrap作为网页技术支持。Bootstrap是流行的HTML、CSS、JS框架，主要用于开发响应式布局，由Twitter的工程师为提高内部效率开发的易用、灵活、可扩展的前端工具集；Jinja2是为Flask开发的基于Python的前端页面模板引擎，设计思想采用Django的模板引擎，轻量化与Flask的微框架搭配使用作为前端视图展现层。

### 5.1.4 推荐系统采用技术方案

推荐系统采用基于用户的协同过滤和基于项的协同过滤互相补充的混合型推荐算法，并通过建立缓存提高推荐效率。

## 5.2 需求分析设计



### 5.2.1电影视频播放网站首页

个人用户页面：用户个人的信息管理页面，负责用户个人信息管理

网站首页：网站首页会根据用户来展示不同的推荐电影

视频播放页面：视频播放页面，播放页面具有电影详细信息、评分、评论区

搜索结果页面：展示用户搜索电影的后的结果

### 5.2.2.用户个人信息管理

用户登录、用户注册：用户登录与注册功能

修改信息：用户修改个人信息、登录密码等

播放记录管理：用户删除个人的播放记录

### 5.2.3.用户行为管理

用户播放功能：用户可以查看视频信息、播放视频的功能

用户评论功能：用户可以在电影下方的评论区发表、查看评论的功能

用户评分功能：用户对于电影的详细信息、简介查看，对电影进行打分

### 5.2.4.站内搜索功能

全局搜索：解除了用户查询条件限制，用户可以通过电影名、电影类型、上映时间等进行精确或者模糊查询。

### 5.2.5.推荐功能

用户喜好电影推荐：用户进入网站首页时，系统会给出个性化首页

5.2.6.管理员功能

电影信息管理：电影信息的管理

电影资源管理：电影播放视频资源链接、图片的资源的管理，

电影播放热度统计：分析电影播放记录信息表，对电影播放次数、评分等信息进行统计、并且依照这些信息计算出电影的热度

用户信息查询：管理员对网站平台用户的信息进查询

## 5.3 数据库设计

根据数据实体之间关系，结果MongoDB数据库的特点设立数据集合。

### 5.3.1 用户数据表

用户表数据实体如图5-3-1.1所示



图5-3-1.1

用户数据表字段名、含义数据类型如表5-3-1.2所示

|  |  |  |  |
| --- | --- | --- | --- |
| 用户表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | userId | 用户编号 | int |
| 2 | username | 用户名 | string |
| 3 | name | 真实姓名 | string |
| 4 | pwd | 密码 | string |
| 5 | phone | 电话 | string |
| 6 | email | 邮箱 | string |
| 7 | other | 备注 | string |
| 8 | status | 删除标志 | int |
| 9 | \_id | MongoDB标识符 | ObjectId |

表 5-3-1.2

### 5.3.2 电影数据表

电影数据实体如图5-3-2.3所示



图5-3-2.3

用户数据表字段名、含义数据类型如表5-3-2.4所示

|  |  |  |  |
| --- | --- | --- | --- |
| 电影表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | movieId | 电影编号 | int |
| 2 | title | 电影名 | string |
| 3 | time | 上映时间 | string |
| 4 | genres | 电影类别 | string |
| 5 | describe | 电影详情 | string |
| 6 | imdbId | 详情链接 | string |
| 7 | tmdbId | 详情链接 | String |
| 8 | vide\_url | 视频资源链接 | String |
| 9 | img\_url | 图片链接 | String |
| 10 | status | 删除标志 | int |
| 11 | \_id | MongoDB标识符 | ObjectId |

表5-3-2.4

### 5.3.3 电影播放记录表

播放记录实体（即训练用数据集）如图5-3-3.5



图5-3-3.5

播放记录字段详情如表5-3-3.6

|  |  |  |  |
| --- | --- | --- | --- |
| 播放记录表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | userId | 用户编号 | int |
| 2 | movieId | 电影编号 | int |
| 3 | ratings | 评分 | int |
| 4 | title | 电影名 | string |
| 5 | comment | 评价 | String |
| 6 | status | 删除标志 | int |
| 7 | \_id | MongoDB标识符 | ObjectId |

表5-3-3.6

### 5.3.4 缓存数据表

为了提高系统运行效率，使用缓存数据表来提高效率

1. 电影相似性矩阵，是相似性算法中的临时数据存储表，存储电影之间的相似性，数据实体如图5-3-4.7，电影相似性矩阵数据字段，如表5-3-4.7。



图5-3-4.7

|  |  |  |  |
| --- | --- | --- | --- |
| 电影相似性矩阵表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | movieId | 电影编号 | int |
| 2 | simi\_movie | 相似电影 | Object |
| 3 | status | 删除标志 | int |
| 4 | create\_time | 生成时间 | date |
| 5 | \_id | mongodb标识符 | ObjectId |

图5-3-4.8

1. 用户相似性矩阵，基于用户的协同过滤推荐算法中用户相似性矩阵的临时缓存表，数据表实体如图5-3-4.9，用户相似性矩阵数据字段，如表5-3-4.10



图5-3-4.9

|  |  |  |  |
| --- | --- | --- | --- |
| 用户相似性矩阵表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | userId | 用户编号 | int |
| 2 | simi\_user | 相似用户 | Object |
| 3 | status | 删除标志 | int |
| 4 | create\_time | 生成时间 | date |
| 5 | \_id | mongodb标识符 | ObjectId |

表5-3-4.10

1. 用户推荐电影表，推荐算法中生成的临时缓存表，为了提高推荐系统效率，推荐电影缓存表定时更新，数据表实体如图5-3-4.11，用户相似性矩阵数据字段，如表5-3-4.12



图5-3-4.11

|  |  |  |  |
| --- | --- | --- | --- |
| 用户推荐电影表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | userId | 用户编号 | int |
| 2 | rec\_movie | 推荐电影 | Object |
| 3 | create\_time | 生成时间 | date |
| 4 | status | 删除标志 | int |
| 5 | \_id | MongoDB标识符 | ObjectId |

表5-3-4.12

1. 电影播放次数数据表，统计电影热度的临时缓存表，更具电影播放热度决定热度排行，数据表实体如图5-3-4.13，用户相似性矩阵数据字段，如表5-3-4.14

图5-3-4.13

|  |  |  |  |
| --- | --- | --- | --- |
| 电影播放次数表 | | | |
| 序号 | 数据库字段名 | 中文含义 | 字段类型 |
| 1 | movieId | 用户编号 | int |
| 2 | title | 电影名 | string |
| 3 | count | 电影播放次数 | int |
| 4 | Ratings | 总评分 | int |
| 5 | create\_time | 生成时间 | date |
| 6 | status | 删除标志 | int |
| 7 | \_id | MongoDB标识符 | ObjectId |

表5-3-4.14

## 5.4 本章小结

本章介绍个性化视频网站开发过程中的技术、需求分析和数据库设计，对于网站功能模块进行了概要设计，根据MongoDB数据库设计出文档数据表，本章完善了前期需求，以及数据标准，为下章功能设计奠定基础。

网站设计中采用的算法为基于用户的协同过滤推荐算法和基于项的协同过滤推荐算法结合，将两种算法的推荐结果结合在一起为用户推荐电影。

1. 个性化视频推荐网站的实现

系统实现是对需求分析中的功能模块的基本实现，主要过程有后台功能实现，前端页面和脚本功能的实现，包含数据库配置、系统设计、代码架构设计和模块设计。

## 6.1 环境搭建

系统环境如下所示

|  |  |  |
| --- | --- | --- |
| 开发环境 | | |
| 开发工具 | 开发语言 | Python |
|  | 后端框架 | Flask |
|  | 前端框架 | JinJa2、Bootstrap |
|  | 数据库 | MongoDB |
| 支持工具 | 数据验证 | 正则表达式、MD5加密算法 |
|  | 缓存数据库 | Redis |

## 6.2 用户网站功能实现

### 5.2.1电影视频播放网站网页设计

|  |  |
| --- | --- |
| 登录网页 | 登录页面 |
|  | 注册页面 |
|  | 修改密码页面 |
|  | 修改用户信息 |
| 主页 | 个人首页 |
|  | 播放页面 |
|  | 搜索页面 |
|  | 查找页面 |
|  | 播放记录页面 |
| 后台管理 | 用户信息管理 |
|  | 电影信息管理 |
|  | 电影信息统计 |

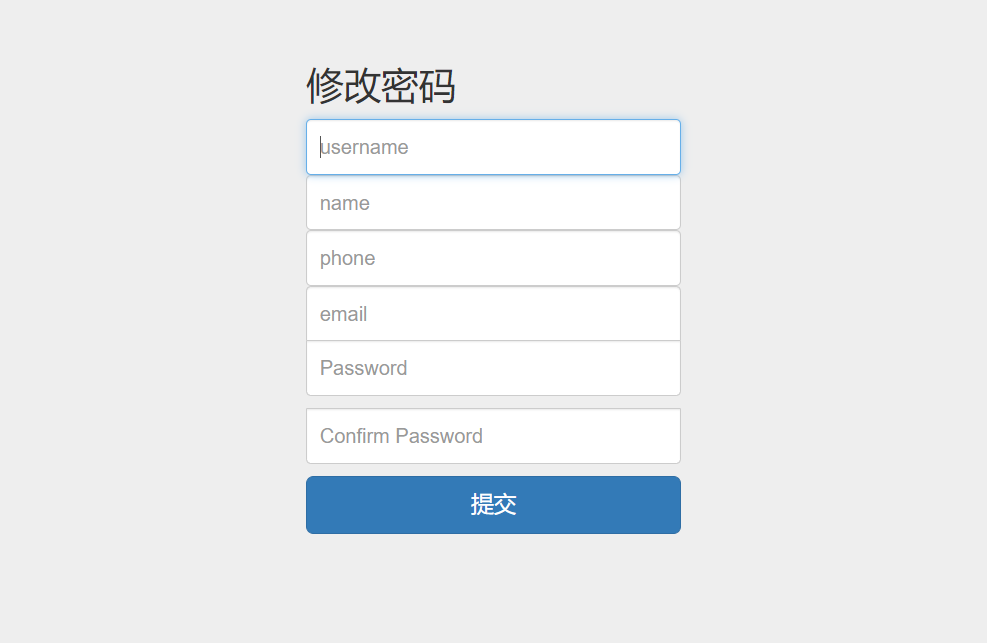
### 5.2.2.用户个人信息管理

用户登录、用户注册等功能：由于HTTP协议的无状态性，采用Session 和Cookies 来进行记录用户的登录状态，为防止Cookies欺骗、采用Token令牌验证方式，使用MD5加密算法来生成Token令牌，服务器存储在Redis缓存中，客户端存储在Cookie中。具体方法为：使用Ptyhon中的hashlib模块，利用用户名、密码、以及加盐固定字符串和登录时间生成用户token信息，token过期时间设置为3小时，用户推出登录时删除token。

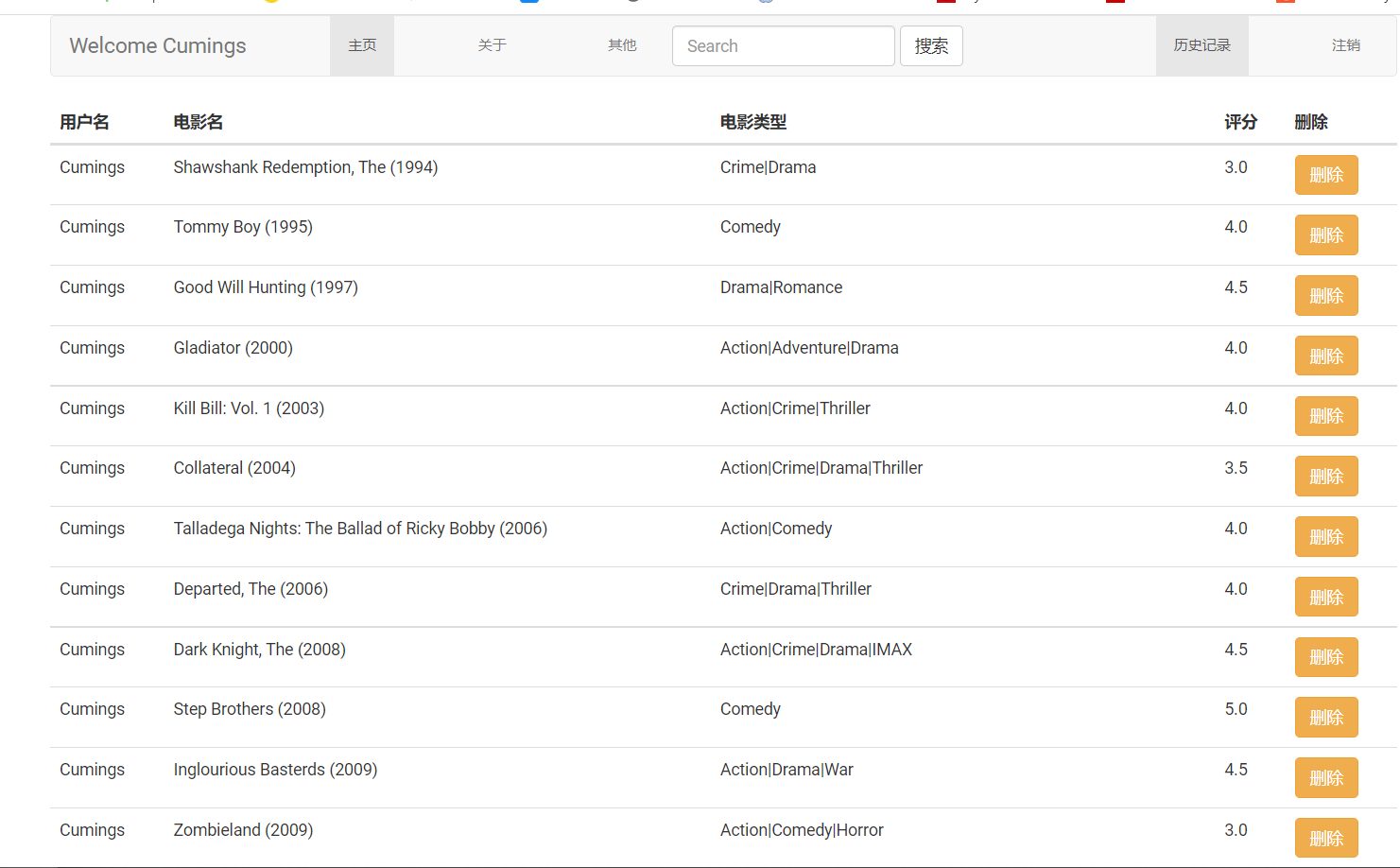




修改信息：用户修改个人信息、登录密码等

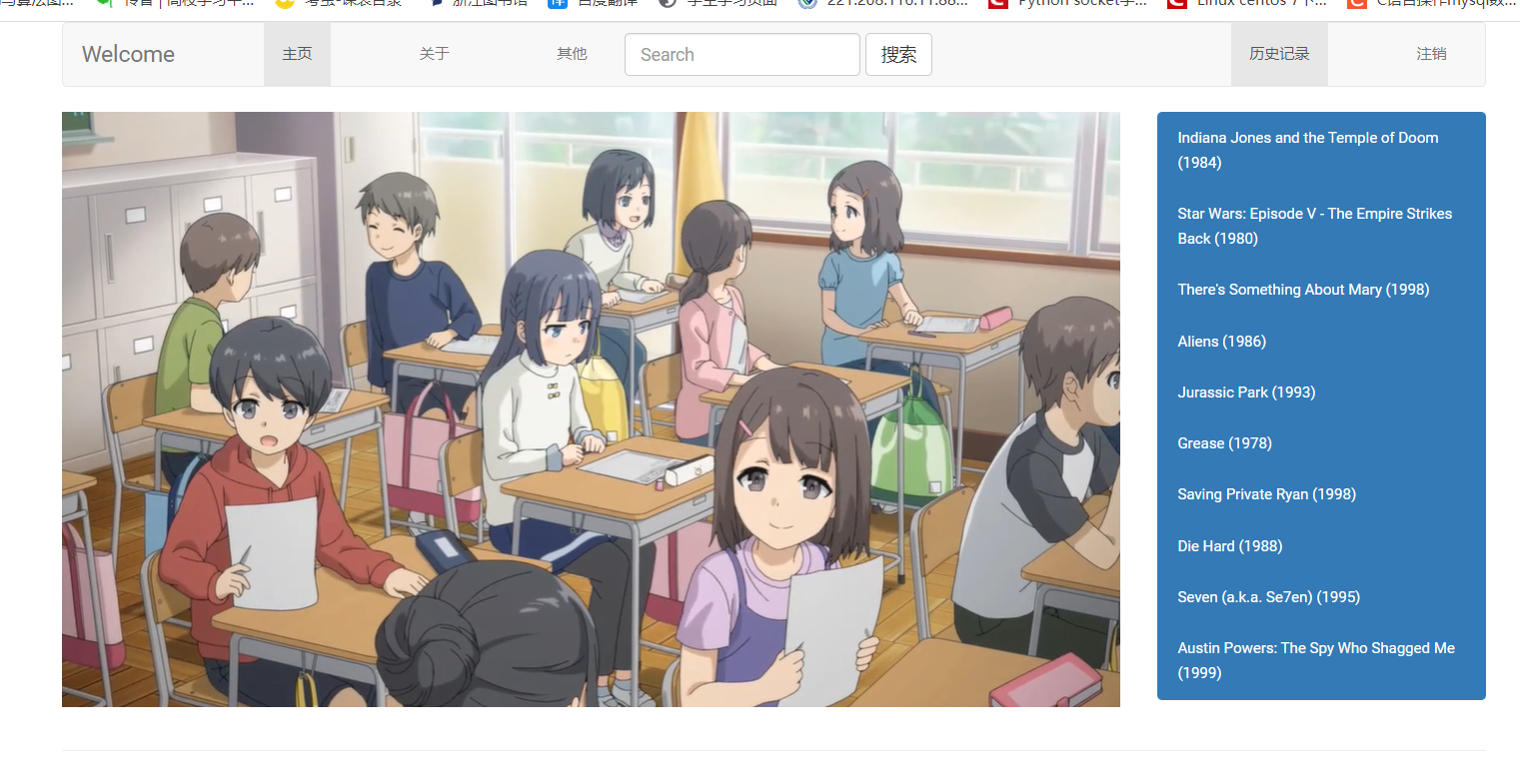


播放记录管理：用户删除个人的播放记录



### 5.2.3.用户行为管理

用户播放功能：用户可以查看视频信息、播放视频的功能

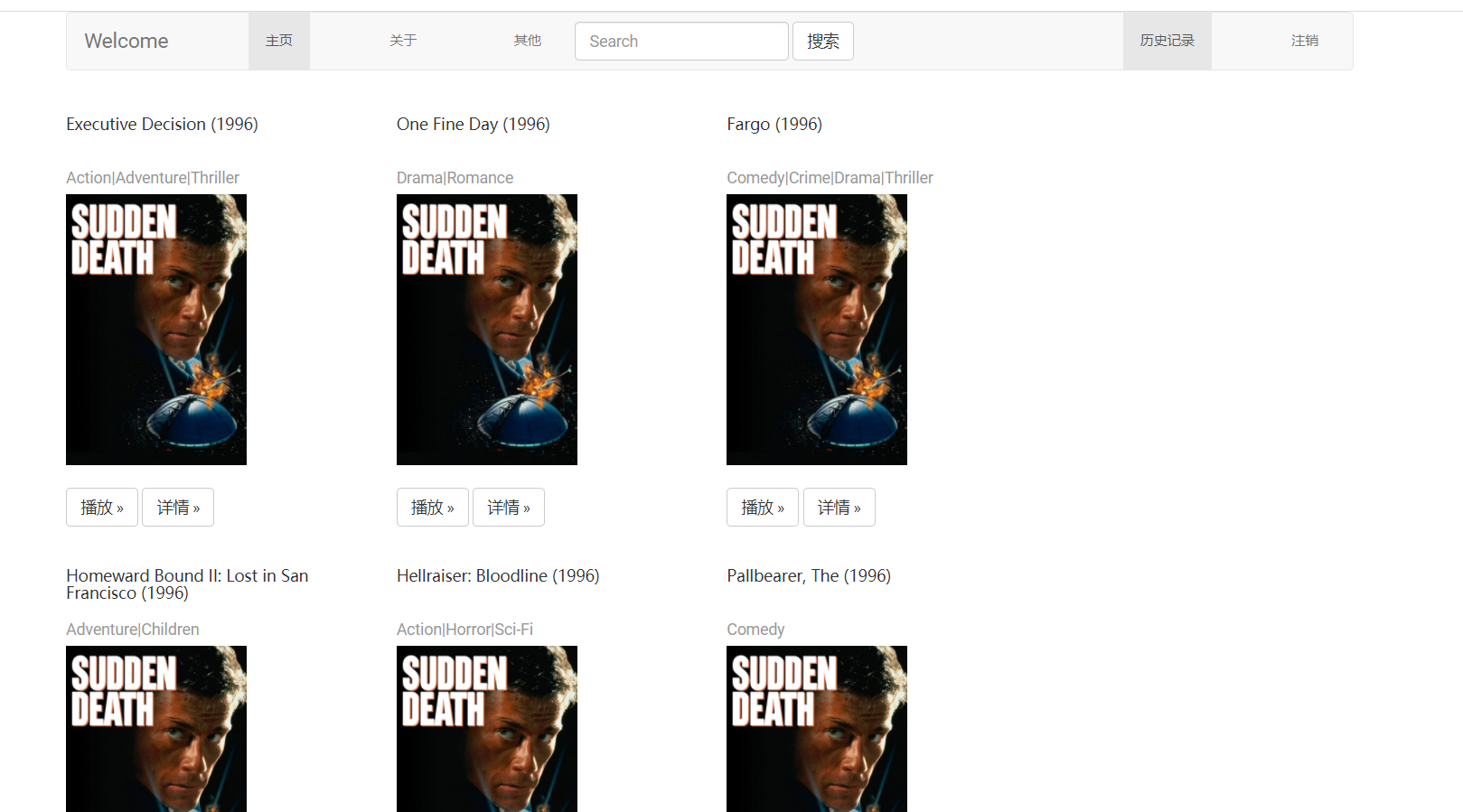


用户评论功能：用户可以在电影下方的评论区发表、查看评论的功能

用户评分功能：用户对于电影的详细信息、简介查看，对电影进行打分

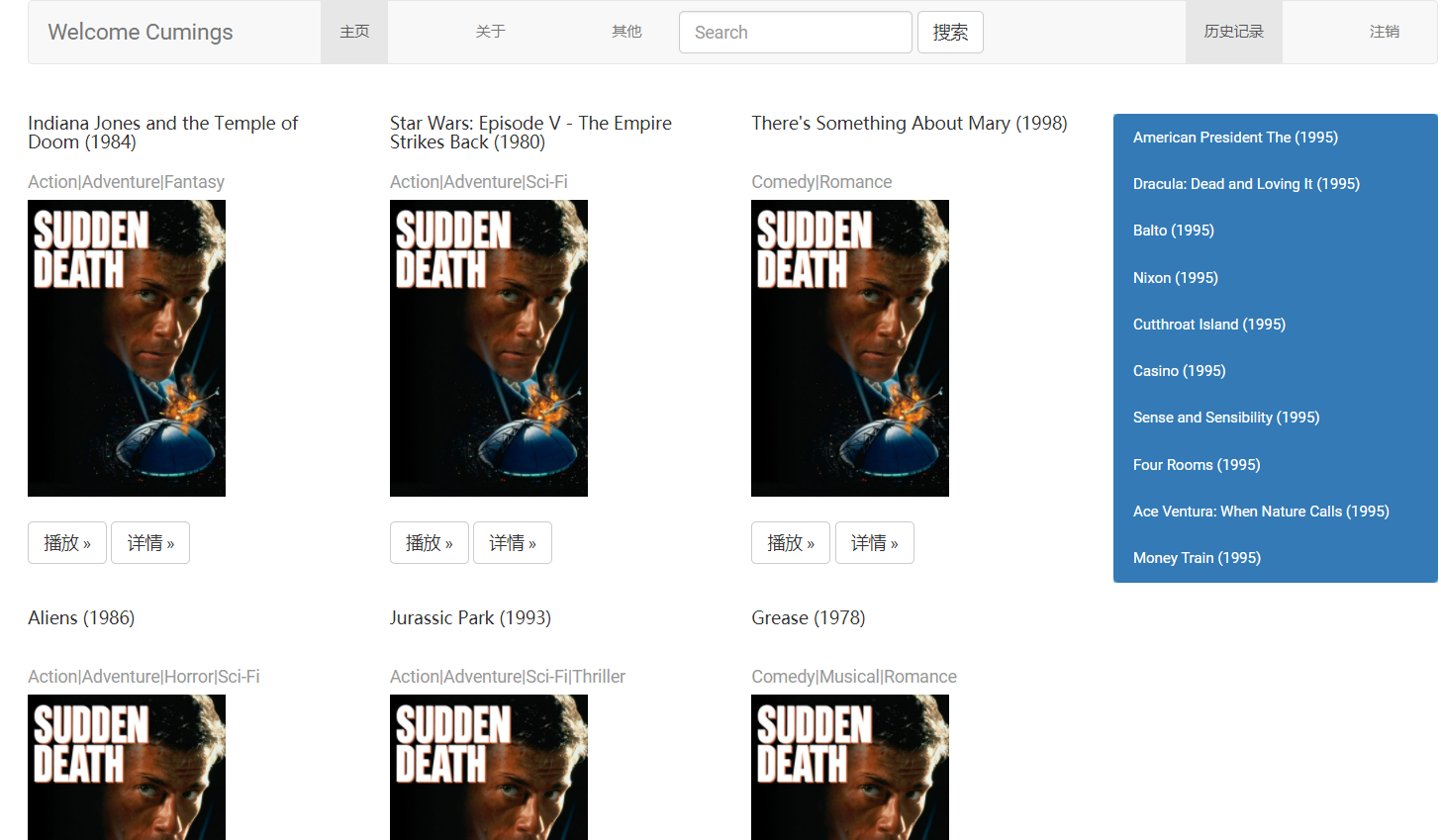
### 5.2.4.站内搜索功能

全局搜索：解除了用户查询条件限制，用户可以通过电影名、电影类型、上映时间等进行精确或者模糊查询，利用MongoDB文档型数据，利用查询条件在所有符合字段里面进行查询。



### 5.2.5.推荐功能

用户喜好电影推荐：用户进入网站首页时，系统会给出个性化首页

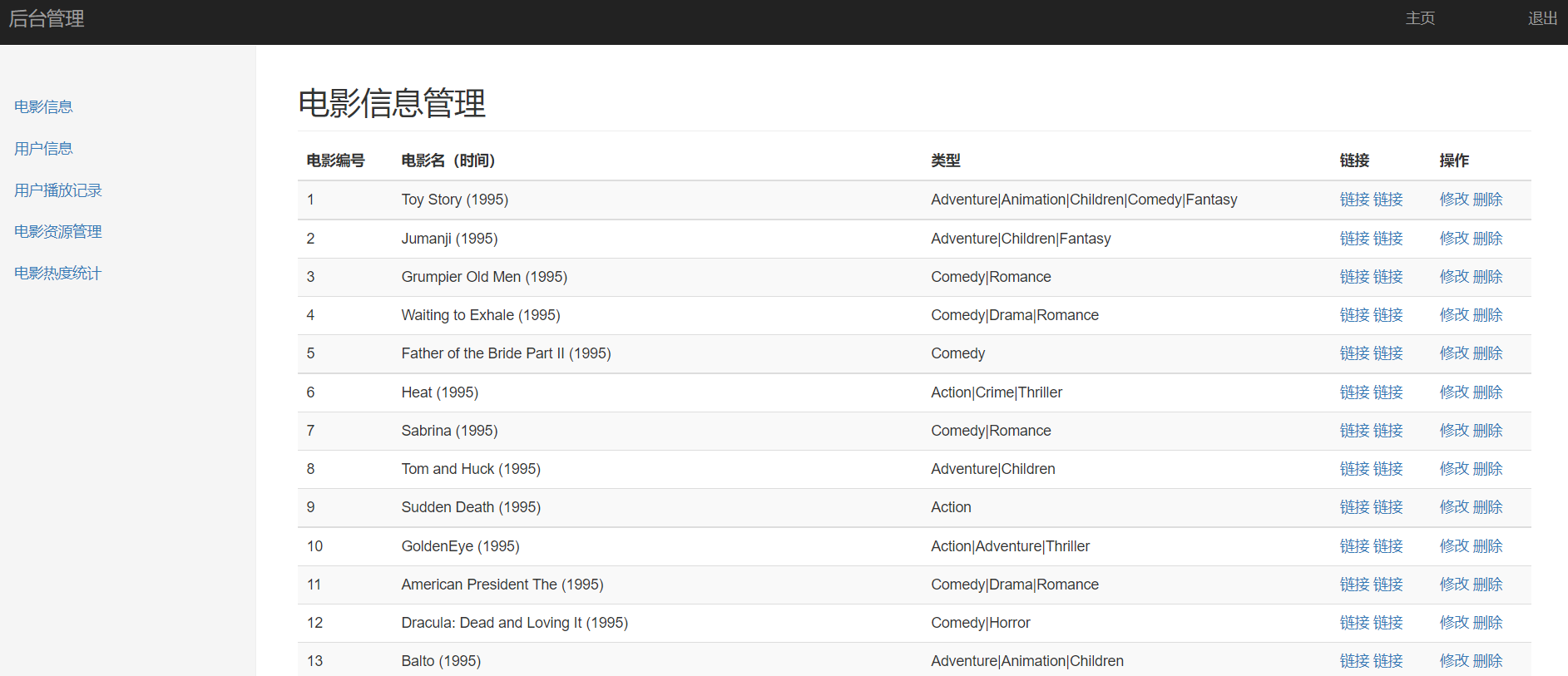


## 

## 6.3 后台管理功能实现

### 6.3.1电影信息管理

对于电影信息的管理修改功能



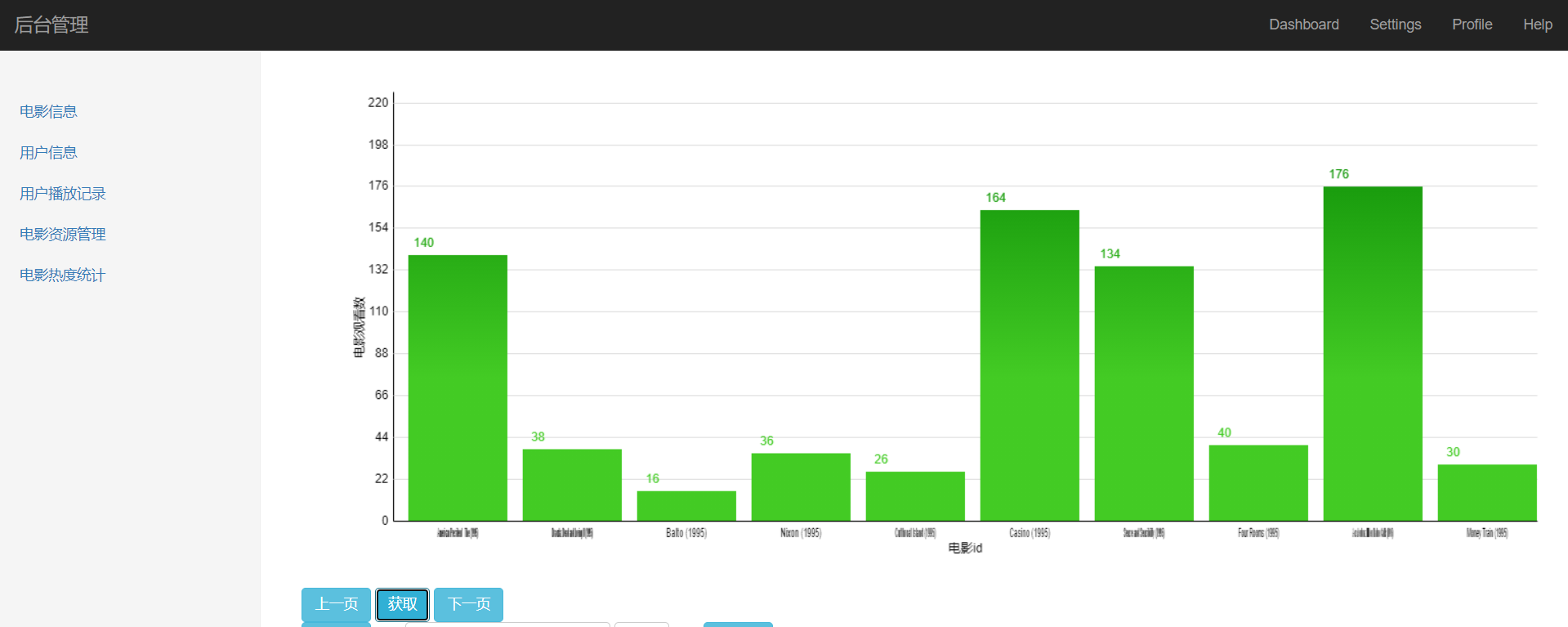
### 6.3.2电影资源管理：

电影播放视频资源链接、图片的资源的管理。



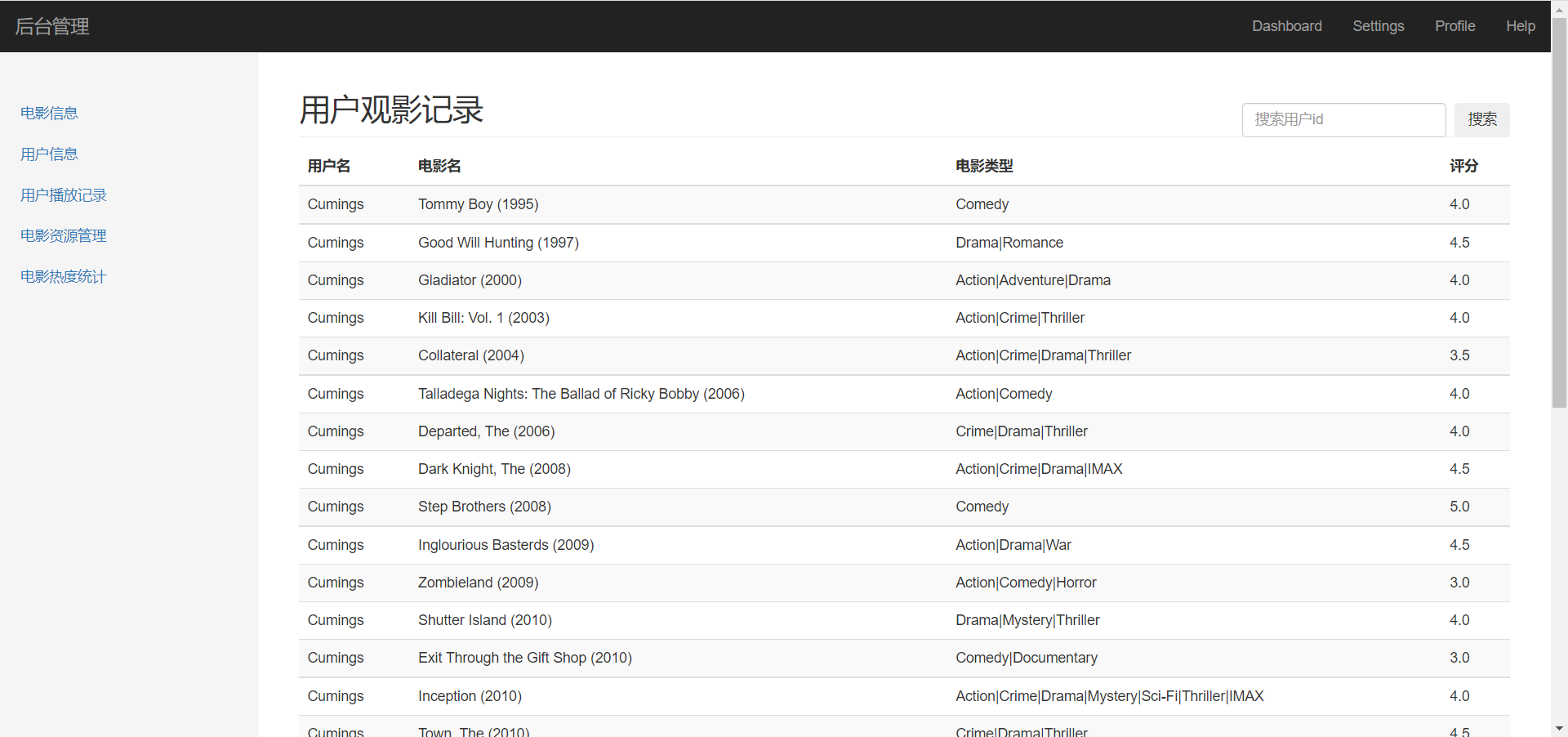
### 6.3.3电影播放热度统计

分析电影播放记录信息表，对电影播放次数、评分等信息进行统计、并且依照这些信息计算出电影的热度



### 6.3.4用户信息管理

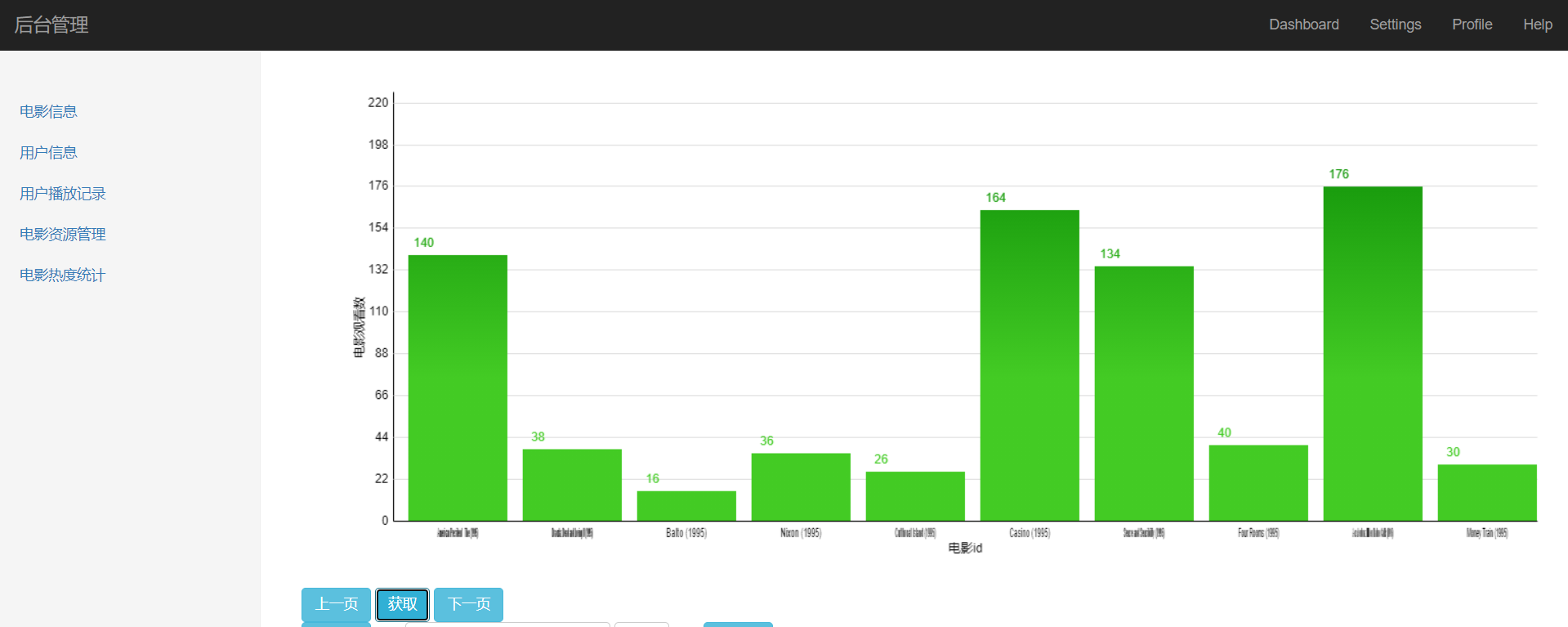
管理员对用户的信息、播放记录查询，为保护用户隐私，只允许查询操作



## 6.4 网站前端首页功能实现

### 6.4.1前端柱形图绘制

前端使用 js技术操作canvas画布，绘制柱形图，数据利用Ajax从后台查询，可以直观的了解电影的播放次数



### 6.4.2前端分页功能实现

分页功能通过物理分页实现操作，减少单次请求的数据量，在低访问的环境下具有最快效果。

## 6.5 本章小结

本章主要对系统网站功能模块进行实现，和运行测试展示，以及项目中一些特殊情况的处理，在项目测试过程中发现问题修改问题。

第7章 总结与展望

## 7.1论文工作总结

当今飞速发展的互联网行业，带来的是大量数据的泛滥，在这个信息爆炸的时代，用户很难从海量的信息中找到自己真正感兴趣的东西，内容的提供商也难以将自己的生成信息推荐到真正感兴趣的用户面前，即我找不到我想要的东西，我想要的东西找不到我，这是一个互相选择的过程，唯一的区别在于，对于生产者，被选择意味着有经济效益，可以将经济效益用于再生产达到企业的发展过程。作为协同消费者（用户）与生产者（内容提供商）之间互相选择的中间服务，推荐系统也由此发展，通过不断学习用户的行为数据，挖掘用户的喜好；同时分析物品的相似性，归类物品分类，为物品找到合适的买家，推荐系统往往是由内容提供商来开发生成，为了提高用户对内容的喜好，利用本平台的数据来进行推荐，增强用户对平台的依赖性，提高平台的市场占有率。其中在各种推荐系统的算法中，协同过滤推荐算法已经被成功应用在亚马逊、YouTube、Hulu等知名网站中。

## 7.2 论文展望

本论文基于杰卡德相似度的协同过滤算法，主要设计到算法的验证挑选实验，以及具体到网站上的应用，属于技术应用型，在使用推荐算法中遇到了较多的问题，局限于算法、硬件的条件导致的速度不足，使用异步式、Redis缓存来减少网站的响应时间，利用用户的观影时间来进行推荐清单的计算。

协同过滤推荐算法是应用非常广泛的算法，本论文在实验中发现基于余弦的推荐算法的计算量是非常庞大，于是对推荐系统的原理更加向往，推荐领域的算法有着非常广阔的研究空间，目前推荐系统大多数依赖于获取用户的信息和行为数据，因此该过程设计大量用户的隐私，在此过程中容易触发侵犯用户隐私的问题，在互联网中，保护用户的隐私是一个比较重要的问题。

在推荐系统中，设计到用户都是数以亿计，怎么样提高推荐系统的效率是需要研究的方向，如何利用分布式平台高效、准确地计算用户日志中的行为数据，建立一个完善的推荐模型。而且协同过滤推荐模型中较为热门的商品会被较多的用户关注，而新商品往往因为缺少数据，有一个不够准确的推荐，因此需要我们去解决推荐系统的单一性。

参考论文

[1]朱扬勇, 孙婧. 推荐系统研究进展[J]. 计算机科学与探索, 2015, 9(5):513-525.

[2]王立才, 孟祥武, 张玉洁. 上下文感知推荐系统[J]. 软件学报, 2012(01):1-20.

[3]刘建国, 周涛, 郭强,等. 个性化推荐系统评价方法综述[J]. 复杂系统与复杂性科学, 2009(03):1-10.

[4]王国霞, 刘贺平. 个性化推荐系统综述[J]. 计算机工程与应用, 2012.

[5]许海玲, 吴潇, 李晓东,等. 互联网推荐系统比较研究[J]. 软件学报, 2009, 20(002):350-362.

[6]刘建国, 周涛, 汪秉宏. 个性化推荐系统的研究进展[J]. 自然科学进展, 2009, 019(001):1-15.

[7]王敏. 基于协同过滤的电子商务个性化推荐系统设计和实现[D]. 广州大学, 2019.

[8]贾忠涛. 基于协同过滤算法的电影个性化推荐系统设计与实现[J]. 软件导刊, 2015, 14(1):86-86.

[9]孙广东. 基于协同过滤的图书推荐系统设计与实现[J]. 2017.

[10]迟轩. 基于协同过滤算法的推荐系统框架设计与实现[D]. 上海交通大学.

[11]尹作文. 基于混合协同过滤的电子商务推荐系统的研究与应用[D]. 武汉理工大学, 2015.

[12]彭兴. 基于Hadoop的电子商务推荐系统的设计与实现[D]. 闽南师范大学, 2016.

[13]孟琳萱. 基于协同过滤的电商个性化推荐系统的设计与实现[J]. 数字化用户, 2019, 025(032):111,113.

[14]迟轩. 基于协同过滤算法的推荐系统框架设计与实现[D]. 上海交通大学.

[15]尹作文. 基于混合协同过滤的电子商务推荐系统的研究与应用[D]. 武汉理工大学.

[16]朱小强, 张琳. 一种改进的协同过滤推荐算法[J]. 现代计算机:中旬刊, 2015.

[17]董辉, 方晓, 马健,等. 基于情景感知的移动电商用户—项目聚类协同过滤推荐算法[J]. 广西民族大学学报(自然科学版), 2018, 024(002):61-68.

[18]电子商务. 基于模糊推理的web客户需求协同过滤推荐算法. 2011.

[19]李亚欣. 基于协同过滤的电子商务个性化推荐算法研究[D]. 东北财经大学, 2012.

[20]刘庆鹏. 基于协同过滤的电子商务个性化推荐算法研究[D]. 海南大学, 2012.

[21]贺桂和. 基于用户偏好挖掘的电子商务协同过滤推荐算法研究[J]. 情报科学, 2013, 031(012):38-42.

[22]王纪辉, 赵卓宁. 基于协同过滤算法的电子商务网站个性化推荐系统设计[J]. 成都信息工程学院学报, 2007, 22(0z1):161-167.

[23]周春华, 沈建京, 李艳,等. 经典推荐算法研究综述[J]. 计算机科学与应用, 2019, 009(009):P.1803-1813.

[24]赵森. 融合用户偏好和项目关联的协同过滤算法研究[D].吉林大学,2020.

[25] Bouzid M , Bonnefoy D , Lhuillier N , et al. RECOMMENDER SYSTEM[J]. US.

[26] Sarwar B M . Application of Dimensionality Reduction in a Recommender System : A Case Study[C]// Acm Webkdd Web Mining for E-commerce Workshop. 2000.

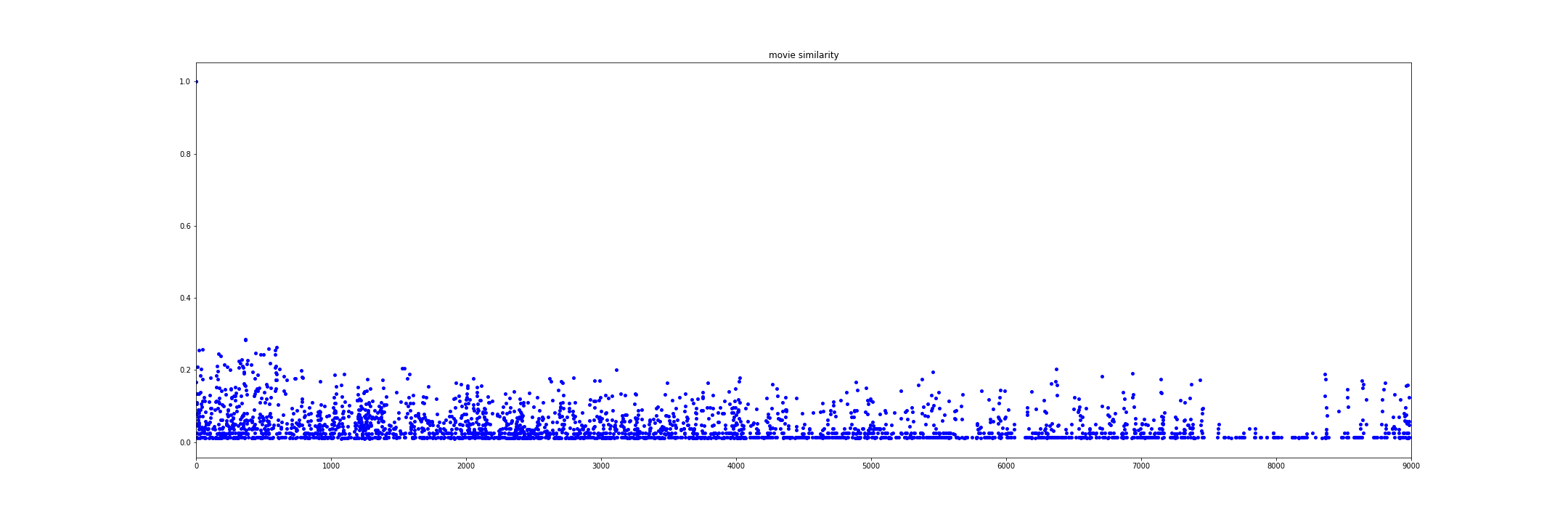
[27] Hema B , Shikha M . A Multi -Perspective Evaluation of MA and GA for Collaborative Filtering Recommender System[J]. International Journal of Computer Science & Information Technology, 2010, 2(5):103-122.

致 谢

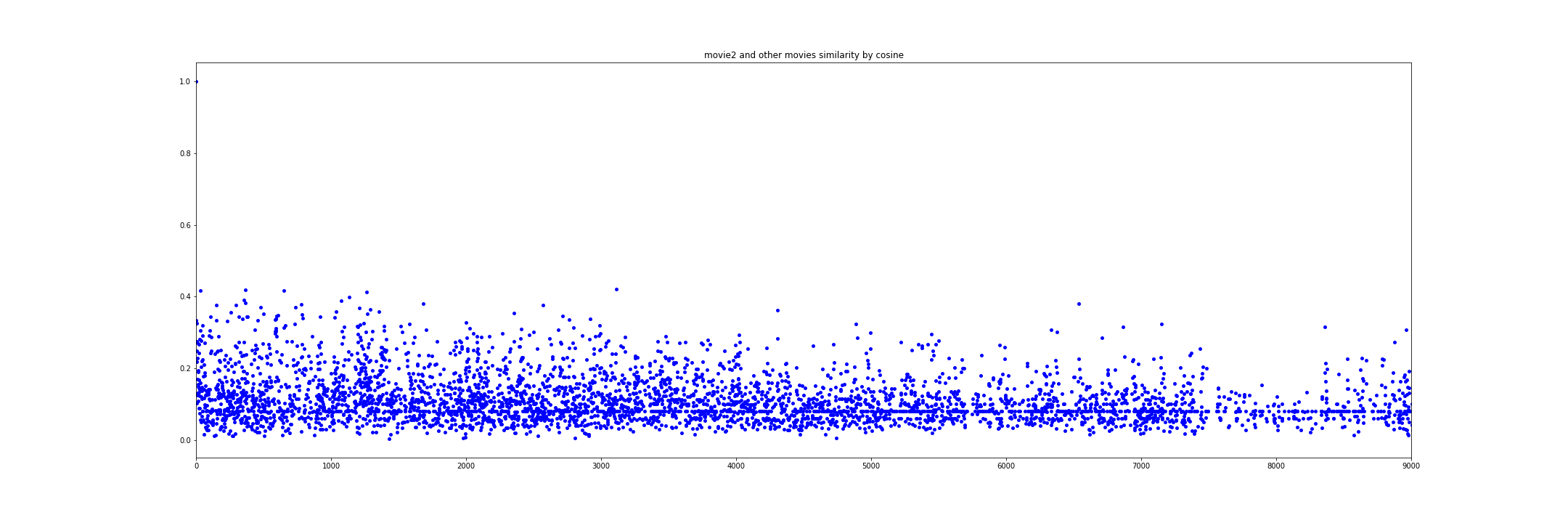
本论文攥写期间遇到很多或大或小的问题，在解决问题的过程深感自己的不足，毕业不是学习的终点也是学习的起点，在解决论文的问题过程中，感谢老师同学之间的帮助，在各位的帮助之下，明确了论文的思路，制定了论文的大纲，在毕设的设计过程中，感谢老师给予的解答，或许老师对于新技术的理解不足，但是宝贵的经验仍然是难得可贵的帮助。

时光荏苒，流年易逝，不知不觉中已经来到毕业，回想四年经历，也算是丰富多彩了，在学习生活中，感谢同学老师给予的帮助，在四年成长历程中、收获了很多，人生漫漫，有还有二十多个四年，希望在未来的时间里，可以砥砺前行、不负青春。

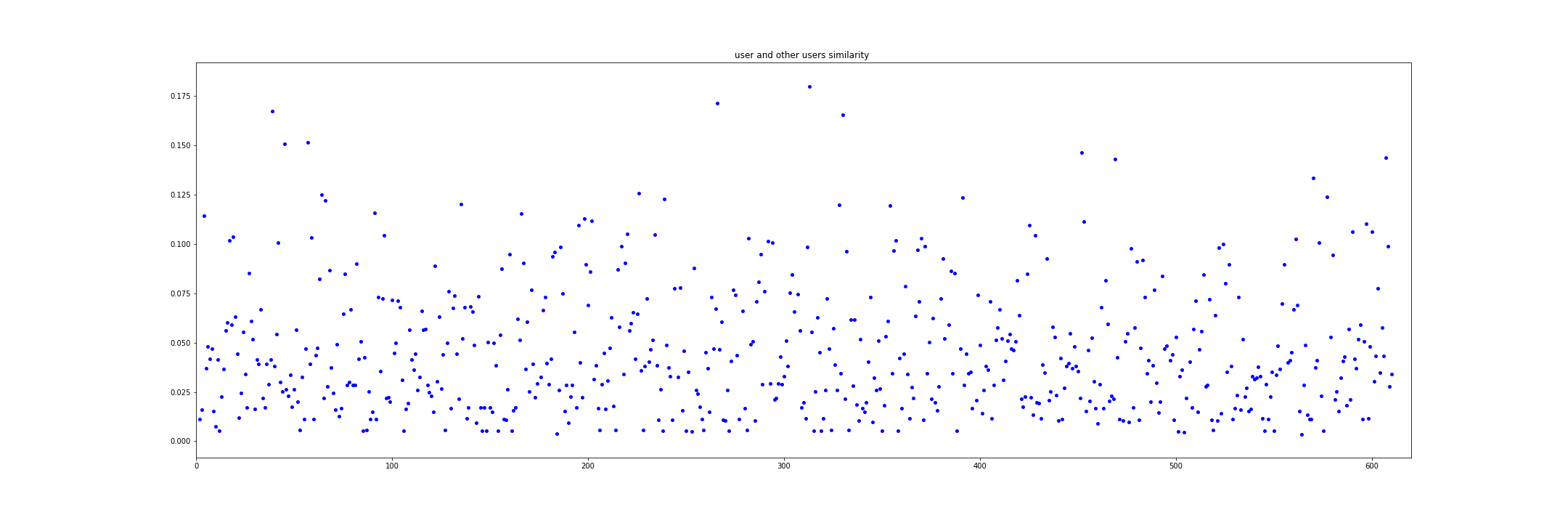
附 录



电影1与其他用户相似度（杰卡德相似度）



电影1与其他用户相似度（余弦相似度）



用户1与其他用户的相似度